WestVirginiaUniversity
THE RESEARCH REPOSITORY @ WVU

Graduate Theses, Dissertations, and Problem Reports

2019

Domain Adaptation and Privileged Information for Visual Recognition

Saeid Motiian

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Recommended Citation

Motiian, Saeid, "Domain Adaptation and Privileged Information for Visual Recognition" (2019). *Graduate Theses, Dissertations, and Problem Reports*. 6271.
https://researchrepository.wvu.edu/etd/6271

This Dissertation is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Dissertation in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Dissertation has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Domain Adaptation and Privileged Information for Visual Recognition

Saeid Motiian

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Gianfranco Doretto, Ph.D., Chair
Donald A. Adjeroh, Ph.D.
Mark Culp, Ph.D.
Victor Fragoso, Ph.D.
Natalia A. Schmid, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2018

Keywords: Domain Adaptation, Privileged Information, Auxiliary Data, Information Bottleneck, Siamese Network, Deep Learning, Adversarial Learning

Copyright © 2018 Saeid Motiian

Abstract

Domain Adaptation and Privileged Information for Visual Recognition

Saeid Motiian

The automatic identification of entities like objects, people or their actions in visual data, such as images or video, has significantly improved, and is now being deployed in access control, social media, online retail, autonomous vehicles, and several other applications. This visual recognition capability leverages supervised learning techniques, which require large amounts of labeled training data from the target distribution representative of the particular task at hand. However, collecting such training data might be expensive, require too much time, or even be impossible. In this work, we introduce several novel approaches aiming at compensating for the lack of target training data. Rather than leveraging prior knowledge for building task-specific models, typically easier to train, we focus on developing general visual recognition techniques, where the notion of prior knowledge is better identified by *additional information*, available during training. Depending on the nature of such information, the learning problem may turn into *domain adaptation* (DA), *domain generalization* (DG), *leaning using privileged information* (LUPI), or *domain adaptation with privileged information* (DAPI).

When some target data samples are available and additional information in the form of labeled data from a different source is also available, the learning problem becomes domain adaptation. Unlike previous DA work, we introduce two novel approaches for the few-shot learning scenario, which require only very few labeled target samples, and even one can be very effective. The first method exploits a Siamese deep neural network architecture for learning an embedding where visual categories from the source and target distributions are semantically aligned and yet maximally separated. The second approach instead, extends adversarial learning to simultaneously maximize the confusion between source and target domains while achieving semantic alignment.

In complete absence of target data, several cheaply available source datasets related to the target distribution can be leveraged as additional information for learning a task. This is the domain generalization setting. We introduce the first deep learning approach to address the DG problem, by extending a Siamese network architecture for learning a representation of visual categories that is invariant with respect to the sources, while imposing semantic alignment and class separation to maximize generalization performance on unseen target domains.

There are situations in which target data for training might come equipped with additional information that can be modeled as an *auxiliary view* of the data, and that unfortunately is not available during testing. This is the LUPI scenario. We introduce a novel framework based on the information bottleneck that leverages the auxiliary view to improve the performance of visual classifiers. We do so by introducing a formulation that is general, in the sense that can be used with any visual classifier.

Finally, when the available target data is unlabeled, and there is closely related labeled source data, which is also equipped with an auxiliary view as additional information, we pose the question of how to leverage the source data views to train visual classifiers for unseen target data. This is the DAPI scenario. We extend the LUPI framework based on the information bottleneck to learn visual classifiers in DAPI settings and show that privileged information can be leveraged to improve the learning on new domains. Also, the novel DAPI framework is general and can be used with any visual classifier.

Every use of auxiliary information has been validated extensively using publicly available benchmark datasets, and several new state-of-the-art accuracy performance values have been set. Examples of application domains include visual object recognition from RGB images and from depth data, handwritten digit recognition, and gesture recognition from video.

Dedication

This thesis is dedicated to my parents for their love, support, and sacrifice. Also to all of my teachers during my student life for their encouragement and motivation.

Acknowledgments

I want to especially thank my PhD advisor, Dr. Gianfranco Doretto, for his great support and all my labmates Quinn, Stanislav, Rania, Sinan, Farzad, Sajid, Harika, and Kerry.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

In the past few years, computer vision technology has reached the attention of the masses because of the widespread use of different cameras, from cellphone to surveillance cameras to more advanced imaging sensors. Computer vision deals with acquiring, processing, and understanding images in order to solve different tasks. Computer vision has a wide range of applications including video gaming [16], in the food industry [17], robotics [18, 19, 20], biomedical [21, 22], and many more [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41].

This dissertation focuses on the visual recognition task. Visual recognition analyses images/videos and provides insights into their visual content. Extensive amount of work has been done in the past to develop robust classifiers that can learn from data how to perform a visual recognition task, from shallow models [42, 43, 44] to deep models [45, 46, 47]. The typical approach requiers collecting enough training samples and their corresponding labels from the target distribution (the one that test samples come from). If we can collect such training pairs, we can obtain very good performance even for very hard tasks (e.g., 1000-class classification task [48]). However, collecting such training pairs might be expensive, impossible, or require too much time. On the other hand, there might be *additional information* that either is already available given the task at hand (*prior knowledge*) or could be cheaply collected to compensate for not having enough training data. For instance, in digit classification, if an image is slightly translated or rotated it still represents the same digit. This prior knowledge indicates that the visual classifier should be invariant to transla-

tions and rotations [49]. [49] reviewed several works using prior knowledge for SVM and we refer the interested reader to [50, 51] for using prior knowledge in deep convolutional networks.

In this study, we focus on different types of additional information that can be collected and used in training. Depending on the nature of such information, the learning problem may turn into several groups:

- **Additional information of the target distribution:** *Supervised learning*, is the machine learning task of learning a model from labeled training data, and is used for visual recognition with promising results [48, 46]. These approaches rely on a fundumental assumption: training (source) data is made of independent identiacal distributed (i.i.d.) samples from the same distribution as the testing (target) data. However, this is an oversimplified scenario because most of the time there is a *covariate shift* [52] (see Figure 1.1) between the *source domain distribution* and the *target domain distribution*. Consider a task where training data and testing data come from slightly different domains (i.e. distributions). In this case plain supervised methods would be suboptimal. The issue could be addressed by collecting *additional information* about the testing/target distribution such as: (1) some unlabeled or very few labeled samples from the target distribution, or (2) more datasets which are closely related to the target distribution.

  In the first case, we could use the available source dataset together with the given target samples to train a model to be robust on the target distribution. This technique is called **domain adaptation** (DA) [53] in the literature. In the second case, the technique would be to use the available source datasets to train a model with good generalization properties on an unknown target domain, which is called **domain generalization** (DG) [54].

- **Additional information of the source distribution:** Sometimes collecting more target data does not necessarily improve the generalization ability of computer vision models [55]. On the other hand, it may be possible to collect some additional information about the available source dataset such as attributes or depth data which will be discussed in Section 1.2.3. In the literature, leveraging this additional information is called **learning using privileged information** (LUPI) [56]. Since privileged/extra information is only available in training, this technique is different from multi-view learning [57, 58] as we will discuss later.

- **Additional information of both source and target distributions:** This can be seen as the combination of DA and LUPI, when both target samples and privileged information is available in training. We call this **domain adaptation with privileged information** (DAPI) [59].

1.2 Motivation and Challenges

In this dissertation, we are looking for learning data representations in an embedding space to address the problems outlined in Section 1.1, and improve the performance of visual recognition on target data. We discuss the above problems by considering two practical examples. First, let's assume that we are interested in developing a cellphone application to recognize objects [60] and we have access to public datasets such as LabelMe [61] and ImageNet [48]. Since testing images come from cellphone cameras, they are slightly different than the training images. Therefore, traditional supervised methods are in a position of weakness. For the second example, imagine if we were interested in training a robot to interact with the environment. Since the scene of a testing environment may be different from training scenes because of lighting, background, etc, traditional supervised methods may perform poorly. In the next sections, we will discuss some techniques to address these examples.

1.2.1 Domain Adaptation

Domain adaptation is one of the most important techniques for solving the covariate shift problem. It is useful whenever we have access only to unlabeled target data or few labeled target data as additional information, because it may be expensive to label or collect them, respectively. In this case, the typical approach is to use available datasets (source data), representative of a closely related task, together with given additional information to train visual recognition models that work well on target domains.

Consider the first example in Section 1.2. Domain adaptation can be useful in this case if we are given some images taken from cellphone cameras in addition to the availabe datasets (LabelMe and ImageNet). Domain adaptation can be either *supervised* [62, 63], *unsupervised* [64, 65], or *semi-supervised* [66, 67, 68].

Unsupervised domain adaptation (UDA) is attractive because it does not require target data to be labeled. Conversely, supervised domain adaptation (SDA) requires labeled target data. UDA mostly tries to minimize the distribution mismatch between source and target datasets, while SDA takes into account the distribution alignment between corresponding classes. UDA expects large amounts of target data in order to be effective. Moreover, given the same amount of target data, SDA typically outperforms UDA, as we will later explain.

Since UDA tries to minimize source and target distributions without considering the alignment of the same classes and the separation of different classes in the two distributions, it is in a position of weakness compared to SDA. Therefore, especially when target data is *scarce*, it is more attractive to use SDA, also because limited amounts of target data are likely to not be very expensive to label. Although domain adaptation can be used for different applications, for example speech processing [69, 70], this dissertation focuses on domain adaptation for visual recognition. Instead of the term "supervised domain adaptation", we may use "few-shot domain adaptation" in this dissertation to stress the fact that we have very few target samples per class during training.

1.2.2 Domain generalization

Sometimes target data are not available and may come from a lot of different distributions. Consider the robot example in Section 1.2, testing scenes widely vary from case to case. In this situation, domain generalization (DG) could be more helpful. In absence of target data, DG exploits several cheaply available datasets (sources) as additional information, representing different specific but closely related tasks. It then attempts to learn by combining data sources in a way that produces visual classifiers that are less sensitive to the specific target data that will need to be processed. DG leads to training visual classifiers that are more robust to distribution changes. This is challenging because DG needs to throw away nuisances in source distributions and only keeps the relevant information.

1.2.3 Learning Using Privileged Information

Sometimes even if we have enough source data, the performance of visual recognition models may still not be very efficient because of unpredictable characterization of target domains [55]. Imagine a case where some additional information is available in training which is missing in

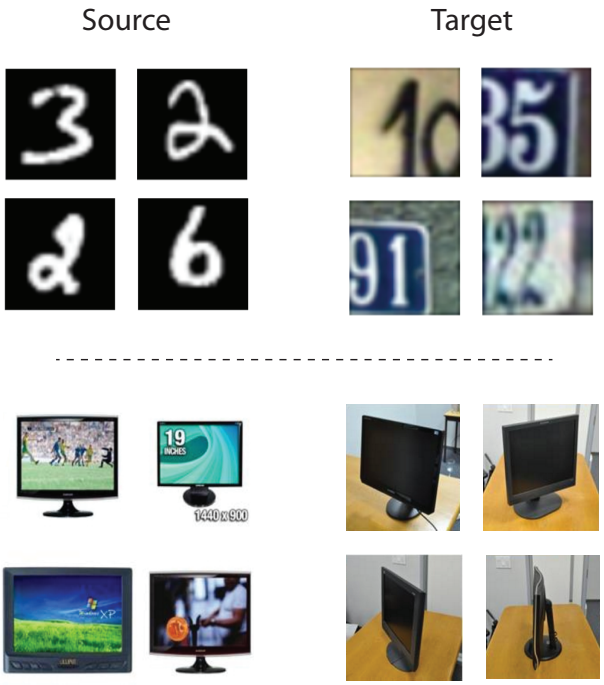

Figure 1.1: Domain Adaptation for Visual Recognition. The final goal is to train a model to work well in the target dataset. However, if there are not enough target samples to train the model, the typical approach is to use available datasets (source data), representative of a closely related task.

testing. Consider the robot example again when we are given additional information in training like depth information of the training scenes, or physical interactions of the robot with the training enviroment [71].

Also consider the cellphone application example, for an object recognition task, a labeled image sample of the *main* data view, representing the *source* domain, might have been annotated also with attributes describing semantic properties of depicted objects, or with a bounding box that specifies the location of the target object, or with image tags describing the context of the image. In testing, only the main view is available because users can only use their cellphones cameras.

Another instance is when dealing with multi-sensory or multimodal data. For example, when processing streams from RGB plus depth sensors, or from multispectral sensors. Visual recognition should leverage data from all the modalities, or data views. However, a view might be missing, possibly due to a sensor malfunction, or to a limited transmission bandwidth, or because we are processing a backlog of historical data where not all the views were recorded.

The main question here is can we train a visual recognition model that is more robust to main data view changes in testing when we are given additional information during training? The answer

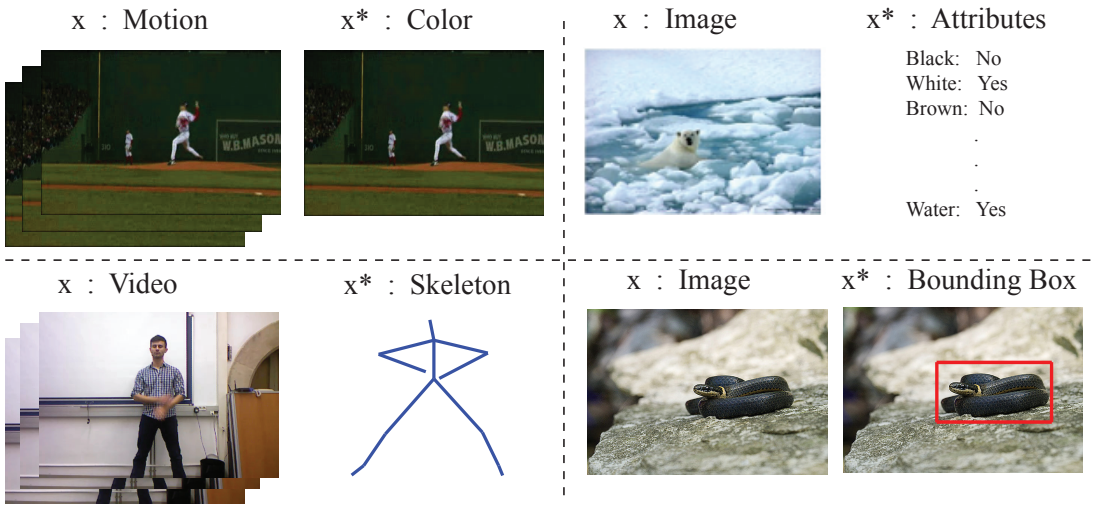

Figure 1.2: Visual recognition with auxiliary data. Visual recognition entails learning classifiers based on a *main* data view (e.g., motion information for recognizing actions, or image information for recognizing animals and objects, or video information for gesture recognition). LUPI tries to leverage an *additional/auxiliary* data view during training (e.g., color for actions, skeleton data for gestures, attributes for animals, and bounding boxes for objects), for learning a better visual classifier.

is "yes". It is possible to learn a shared embedding space using the given views (information) in training which is more robust to main data view changes in testing. Typically, this problem is addressed by processing the available data views with classifiers trained on the same views. However, the missing view at testing time can be seen as additional information, available only during training. This additional information is an *auxiliary* data view of the image/video sample.

This paradigm, improving visual recognition based on a main data view, by leveraging the auxiliary view available only during training, is called Learning Using Privileged Information (LUPI) [56]. See Figure 1.2. LUPI does not minimize the covariate shift between source and target domains, however it leads to models that are more robust to the variability of the target domain.

LUPI has received limited attention. It is different from domain adaptation and transfer learning [72, 13] because of the existence of auxiliary views in training and not existence of target data in training. Indeed, the problem is more related to multi-view and multi-task learning [73, 74, 75, 76, 77]. However, rather than having all views or task labels available or predicted during testing, here one view is missing and a single task label is predicted. The fact that the auxiliary view is missing is what makes this problem challenging, because it cannot be combined like the others in multi-view learning.

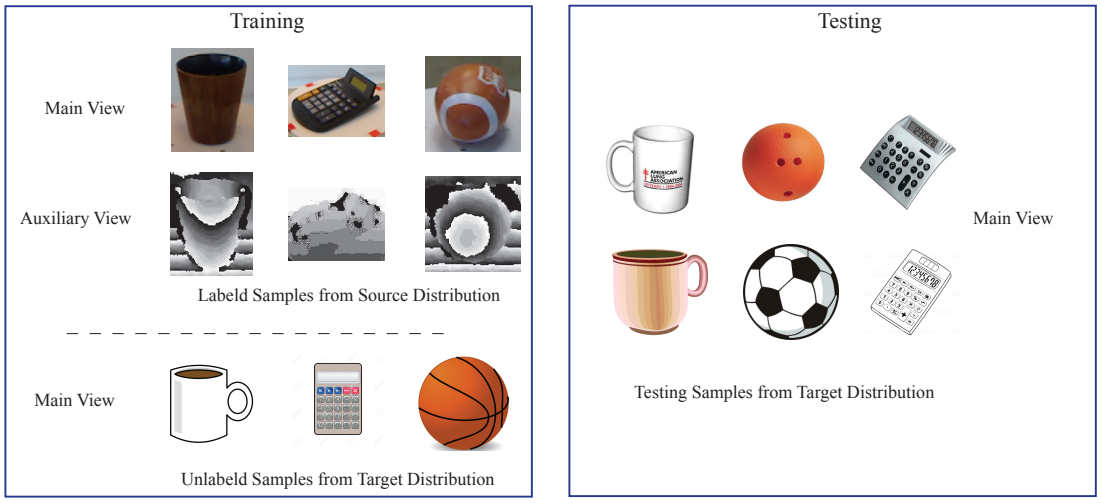

Figure 1.3: Domain Adaptation with Privileged Information. In this scenario, additional information from both source and target distributions are available in training. Since target data distribution and source data distribution differ by a *covariate shift*, the classifier trained only on the main view of the source distribution is suboptimal. Labeled paired source auxiliary/privileged data (e.g., depth data) can be used, along with unlabeled target data, to improve visual recognition on the target domain.

1.2.4 Domain Adaptation with Privileged Information

Imagine the LUPI paradigm while we are also given some labeled or unlabeled target data in training as additional information. Consider the cellphone application example, when in training we are given some images from cellphones together with available source datasets (ImageNet and LabelMe) and some auxiliary views (e.g. object attributes or corresponding object bounding boxes).

The question here is can we minimize the covariate shift while leveraging auxiliary views? The problem outlined above has received very limited attention. It is different from domain adaptation and transfer learning [72], because of the presence of the auxiliary view as part of the source. It is also different from the LUPI paradigm (explained above), because of the presence of main target data in training. Compared to multi-view and multi-task learning [73, 74, 75, 76, 77], instead, rather than having all views or task labels available or predicted during testing, here one view is missing, and a single task label is predicted based on a biased view. Therefore, the asymmetry of the missing auxiliary view already poses a challenge (because it cannot be combined like the others in multi-view learning), which becomes even greater when there is a mismatch between the distributions of the source main view and the target view.

1.3 Contributions and Dissertation Structure

In this dissertation, we introduce algorithms for each of the problems we explained in the previous section. Chapter 2, which is from our ICCV 2017 paper [5], addresses the domain adaptation and domain generalization problems. Chapter 3 proposes a domain adaptation method based on adversarial learning and draws from our NIPS 2017 paper [78]. Chapter 4 addresses the LUPI problem and comes mostly from our CVPR 2016 paper [79]. Chapter 5 draws significantly from our work published in ECCV 2016 [80] and addresses the domain adaptation with privileged information problem.

1.3.1 Chapter 2

In general, domain adaptation methods attempt to minimize the domain shift between source and target domains using three different strategies. The first one tries to find a mapping between source and target distributions [13, 81]. In testing, a target image is first mapped to the source distribution and is then passed to a pre-trained model on source data. The second strategy seeks to learn embedding functions to a shared latent space for source and target distributions [82, 83]. Computer vision models will be trained using source data (and also labeled target data if available) in the shared latent space. The third strategy mainly focuses on regularizing a classifier trained on a source distribution to work well on a target distribution [84, 85].

The second strategy is more popular because it can leverage deep models with a siamese structure [86]. Siamese architectures are powerful networks to find the shared latent space for different tasks and have been used before for domain adaptation [4]. Specifically, unsupervised domain adaptation methods with siamese structures attempt to minimize the covariate shift between source and target domains. To do so, covariate shifts between two domains are often measured with the Maximum Mean Discrepancy (MMD) [3] criteria. MMD is popular because it can be used in stochastic gradient optimization and can be easily combined with kernels. Supervised domain adaptation methods [62] with siamese structures mostly find a shared embedding space such that the distances between corresponding classes in source and target domains are minimum, because they have access to the labeled target data. This leads to better performance compared to unsupervised methods.

In this chapter, we assume a scenario where only very few target labeled data per class is

available in training. This supervised setting becomes more attractive than the unsupervised setting because of the presence of labeled data and when collecting large amount of data is not feasible. We are also interested in siamese networks. Our main challenge is having scarce target data. Therefore, we cannot effectively compute distances between corresponding class distributions in source and target domains. Another challenge is the separation between different classes in the shared embedding space, which is missing in the domain adaptation literature.

Therefore, the main idea in this chapter is to exploit the *siamese architecture* to learn an embedding subspace that is discriminative, and where mapped visual domains are *semantically aligned* and yet *maximally separated*. Since alignment and separation of semantic probability distributions is difficult because of the lack of data, we found that by reverting to *point-wise surrogates* of distribution distances and similarities provides an effective solution. In addition, the approach has a high speed of adaptation, which requires an extremely low number of labeled target training samples, even one per category can be effective. We also extend our proposed model to domain generalization. For both applications the experiments show very promising results.

1.3.2 Chapter 3

In this chapter, we proposed a method for addressing the supervised domain adaptation using adversarial learning [87] for the first time in the literature. This is important because adversarial learning has shown promising results in different tasks including unsupervised domain adaptation [88, 89, 90]. Here we use adversarial learning to train networks such that embedded samples from different distributions are not distinguishable. We consider the task where very few labeled target data are available in training. With this assumption, it is not possible to use the standard adversarial loss used in [88, 89, 90], because the training target data would be insufficient. We address that problem by modifying the usual pairing technique used in many applications such as learning similarity metrics [86, 91, 92] and domain adaptation (Chapter 2). Our pairing technique encodes domain labels as well as class labels of the training data (source and target samples), producing four groups of pairs. We then introduce a multi-class discriminator with four outputs and design an adversarial learning strategy to find a shared feature space. Our method also encourages the semantic alignment of classes, while other adversarial UDA approaches do not.

1.3.3 Chapter 4

We explore the visual recognition problem from a main data view when an auxiliary data view is available during training. This is important because it allows improving the training of visual classifiers when paired additional data is cheaply available, and it improves the recognition from multi-view data when there is a missing view at testing time. The problem is challenging because of the intrinsic asymmetry caused by the missing auxiliary view during testing.

This problem has been addressed before for some specific tasks or for some specific classifiers [56, 93, 10]. One LUPI implementation is the SVM+ [56, 93], which uses the privileged data as a proxy for predicting the slack variables. The same idea has also been used in the learning to rank approach introduced in [10].

Our approach exploits the privileged information differently. We account for privileged information during training by extending the information bottleneck (IB) method [8], and by combining it with risk minimization. This information theoretic framework learns how to compress the source domain for doing prediction in a way that is as informative of the privileged source domain as possible, regardless of the type of classifier used, and without tying privileged information to slack variables. We use this principle to design a large-margin classifier with an efficient optimization in the primal space. We extensively compare our method with the state-of-the-art on different visual recognition datasets, and with different types of auxiliary data, and show that the proposed framework has a very promising potential.

1.3.4 Chapter 5

We address the unsupervised domain adaptation problem for visual recognition when an auxiliary data view is available during training. This is important because it allows improving the training of visual classifiers on a new target visual domain when paired additional source data is cheaply available. The problem is challenging because of the intrinsic asymmetry caused by the missing auxiliary view during testing and from which discriminative information should be carried over to the new domain. We jointly account for the auxiliary view during training and for the domain shift by extending the information bottleneck method, and by combining it with risk minimization. In this way, we establish an information theoretic principle for learning any type of visual

classifier under this particular settings. We use this principle to design a multi-class large-margin classifier with an efficient optimization in the primal space. We extensively compare our method with the state-of-the-art on several datasets, by effectively learning from RGB plus depth data to recognize objects and gender from a new RGB domain.

The only work addressing the same problem as ours is [59], and extended in [94] for web data. They jointly learn a multiclass large-margin classifier, as well as two projections for the main and the auxiliary views, respectively. This is done while maximizing the correlation among views, as well as minimizing the distribution mismatch according to the MMD. On the other hand, we extend the IB method into a general principle that handles the auxiliary view as well as the distribution mismatch from a single information theoretic point of view. Computationally, this entails the estimation of only one projection, rather than two. It allows handling source data points with missing auxiliary view, and we also provide an implementation of a large-margin multiclass classifier in the primal space for improved computational efficiency.

1.4 Related work

Domain Adaptation. Visual recognition algorithms are trained with data from a *source domain*, and when they are tested on a *target domain* with marginal distribution that differs from the source, we experience the visual domain adaptation (DA) problem (also known as dataset bias [95, 55, 96], or covariate shift [52]), and observe a performance decrease. Let us assume that we are given a training dataset made of pairs $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^N$. The feature $x_i^s \in \mathcal{X}$ is a realization from a random variable X^s, and the label $y_i^s \in \mathcal{Y}$ is a realization from a random variable Y^s. In addition, we are also given the training data $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^M$, where $x_i^t \in \mathcal{X}$ is a realization from a random variable X^t, and the labels $y_i^t \in \mathcal{Y}$. At this point y_i^t for some or all t may not be available.

We assume that there is a *covariate shift* [52] between X^s and X^t, i.e., there is a difference between the probability distributions $p_s(x)$ and $p_t(x)$ ($p_s(x) \neq p_t(x)$). The earlier works on domain adaptation have an assumption on conditional distributions: $p_s(y|x) = p_t(y|x)$. This is one of the most widely used assumptions in domain adaptation and corresponds to weight-sharing in deep networks. In the dissertation, we are only interested in $p(y|x)$, which is useful for the classification task. Domain adaptation models can be grouped into three categories:

- **Mapping from source to target.** The first one includes those models that try to find a

mapping between source and target distributions [13, 81, 97, 66, 98, 99]. The earlier works on domain adaptation usually follow this strategy. [52], which first addressed the covariate shift, proposed to re-weight the log-likelihood of each training instance (x, y) using $\frac{p_t(x)}{p_s(x)}$. Let us assume that we are interested in the classification/regression task using the function f, which comes from a set of possible functions F, given some target training pairs. The ultimate goal of a learning algorithm is to find a hypothesis h^* among a fixed class of functions $\mathcal{H}$ for which the risk is minimal. The risk associated with hypothesis $h(x)$ is defined as the expectation of the loss function (ℓ):

$$R(h; L, p_t) = E_{(x,y)\sim p_t}[\ell(f(x), y)] \,, \tag{1.1}$$

where $p_t(x, y)$ is the joint probability over X^t and Y. The samples of the target domain are either too small to provide a reliable approximation of the expected risk, or some or even all of their labels may not be available. Therefore, using the source pairs is the typical approach. Considering our main assumption $(p_s(y|x) = p_t(y|x))$, it is easy to show that (1.1) can be rewritten as:

$$R(h; L, p_t) = E_{(x,y)\sim p_s}[\frac{p_t(x, y)}{p_s(x, y)}\ell(f(x), y)] \,, \tag{1.2}$$

This shows that it is possible to train a visual recognition model for target domain by properly weighting the source samples. The re-weighting function can be rewritten as:

$$\omega(x, y) = \frac{p_t(x, y)}{p_s(x, y)} = \frac{p_t(x)}{p_s(x)}\frac{p_t(y|x)}{p_s(y|x)} = \frac{p_t(x)}{p_s(x)}. \tag{1.3}$$

[52] discussed estimating $p_s(x)$ and $p_t(x)$ using parametric or nonparametric methods. Since density estimation may suffer from the curse of dimensionality [60], it is better to directly learn the weights without estimating the source or the target marginal distributions. Let' s put $\omega(x)p_s(x) = p_t(x)$ and the goal now is to find a weighting function $\omega^*(x)$ that minimize a distance between $\omega(x)p_s(x)$ and $p_t(x)$. Among different distance metrics, Maximum Mean Discrepancy (MMD) [3] has shown very good performance in domain adap-

tation [100, 101, 102]. Empirical MMD can be computed by

$$\text{MMD}_{emp}(X_s, X_t) = ||\frac{1}{N}\sum_{n=1}^{N} k(x_s^n, .) - \frac{1}{M}\sum_{m=1}^{M} k(x_t^m, .)||_{\mathcal{H}} \ , \tag{1.4}$$

where N and M are the total numbers of source and target data and $k(.,.)$ denotes a kernel function associated with the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. MMD is powerful but has limitations. For example, it is hard to find an optimum kernel. Despite some methods which have tried to address some of the issues of MMD such as [60], re-weighting the source data fell out of favor until very recently. In the past, source re-weighting has been done at feature level but due to the rapid development in deep learning, re-weighting can be seen as a mapping from source data to the target data in an original space level (i.e. raw pixels). Let's assume that $x_t = G(x_s)$, where $G(.)$ is a mapping from the source distribution to the target distribution that preserves the content (it is closly similar to $\omega(x)$). If we can find such a mapping function, (1.1) can be rewritten as:

$$R(h; L, p_t) = E_{(x,y)\sim p_s}[\ell(f(G(x)), y)] \ , \tag{1.5}$$

allowing to use the source samples to train a visual recognition model for the target domain. $G(.)$ can be perfectly obtained by using adversarial learning [87] and can perform distributions alignment in raw pixel space, translating source data to the style of a target domain. The challenge here would be how to encourage the model to preserve semantic information during the distribution alignment. This issue arises from the fact that we cannot obtain a pair of (x_s, x_t) in training to learn a good $H(.)$. [103] trains a mapping function to map a source image into a target image by enforcing consistency in the embedding space. [104] uses an L-1 reconstruction loss to force the generated target images to be similar to their original source images. [103, 104] are suitable for tasks with small shift. CycleGAN [105] introduced cycle-consistency to reconstruct the original image from the mapped image. CyCADA [106] adapts representations at both the pixel-level and feature-level while enforcing local and global structural consistency through pixel cycle-consistency and semantic losses. There are some other methods following a similar idea [107, 108]. This category requires enough training

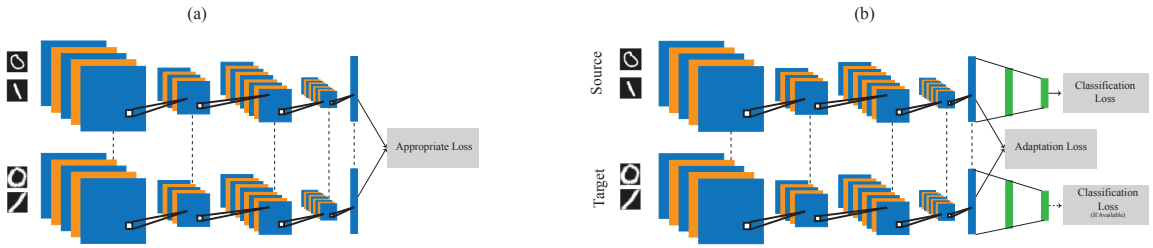

Figure 1.4: Siamese Networks. (a) A siamese network contains two identical sub-networks. Although the sub-network can be implemented by any machine learning model, convolutional neural networks (CNN) are usually used because of their performance. Depending on the task at hand, we can use different loss functions. For the verification task, the Contrastive loss [1] or the Triplet loss [2] are good options. (b) For domain adaptation, several distance metrics can be used in order to minimize the distance between the source and target distributions or samples such as MMD [3], correlation alignment [4], and the Contrastive loss [5]. Also, it is necessary to train the siamese network parameters by jointly minimizing a classification loss and an adaptation loss

target data and therefore is suitable for unsupervised domain adaptation.

- **Finding a shared latent space.** When we do not have access to enough target samples in training, finding a mapping from source to target or estimating the true weights for re-weighting the source samples becomes challenging. Therefore, it is better to map source and target samples to a latent space such that the source and target domains are distributed in the same way (or confused) in that space while the discriminative information has been preserved [83, 109, 110, 111, 79]. If the source and target domains are maximally confused in the latent space, it is safe to assume that $p(y|z)$ is shared by the source and target domains, where $z \in \mathcal{Z}$ represents a feature representation in the shared latent space. In other words, we can use the same classifier, trained on source samples, for target samples which is the **main assumption** made by this category of approaches.

  [112] finds two projections from source and target to the latent space using canonical correlation analysis (CCA). [113] proposes an approach for cross-view action recognition by projecting the action descriptors extracted from source view and those extracted from target view to virtual views in an unsupervised fashion. [114] tries to learn some transfer components across domains in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD) such that data distributions in different domains are close to each other in that space. [82] minimizes the distribution mismatch between the labeled source images and unlabeled target images, and incorporates this criterion into the objective function of sparse coding to make the new representations robust to the distribution difference.

Among algorithms for learning similarity metric, siamese networks [115, 86] work well for different tasks, and when used with deep convolutional neural networks (CNN) [45, 46], they perform well [116, 117]. A siamese network contains two identical sub-networks joined at their outputs (see Figure 1.4). Depending on the type of task at hand, we can use different functions for joining the sub-networks outputs. Recently, siamese networks have been used for domain adaptation. Since we focus on the classification task, the classification loss needs to be added to the siamese network loss, Figure 1.4. Since all the network weights are shared between source and target domains, the conditional distributions are identical $p(Y|X^s) = p(Y|X^t)$. In [62], unlabeled and sparsely labeled target domain data are used to optimize for domain invariance to facilitate domain transfer while using a soft label distribution matching loss.

In [4] (CORAL), which is a deep UDA approach, unlabeled target data is used to learn a nonlinear transformation that aligns correlations of layer activations in deep neural networks. CORAL minimizes the distance between the second-order statistics (covariances) of the source and target samples in feature space:

$$l_{CORAL}(X_s, X_t, F) = \frac{1}{4d^2}||C_S - C_T||_F^2 \,, \tag{1.6}$$

where $||.||_F^2$ denotes the squared Frobenius norm, C_S and C_T denote the feature covariance matrices, and d is the dimension of the feature space. End-to-end domain adaptation can be done by jointly minimizing (1.6) and a classification loss on labeled samples.

Some approaches went beyond the siamase weight-sharing ($p(y|x^s) \neq p(y|x^t)$ but $p(y|z)$ is still shared between two domains) and used coupled networks for DA. [118], which is a deep UDA approach and can be seen as a SDA after fine-tuning, uses a two-streams architecture, for source and target, with related but not shared weights. It consistently outperforms networks with shared weights in the same setting. To better model the shift and introduce more flexibility in the model, [118] proposes an L_2 and an exponential regularizer between the source and target weights:

$$r_w(\theta_j^s, \theta_j^t) = ||a_j\theta_j^s + b_j - \theta_j^t||_2^2 \,, \tag{1.7}$$

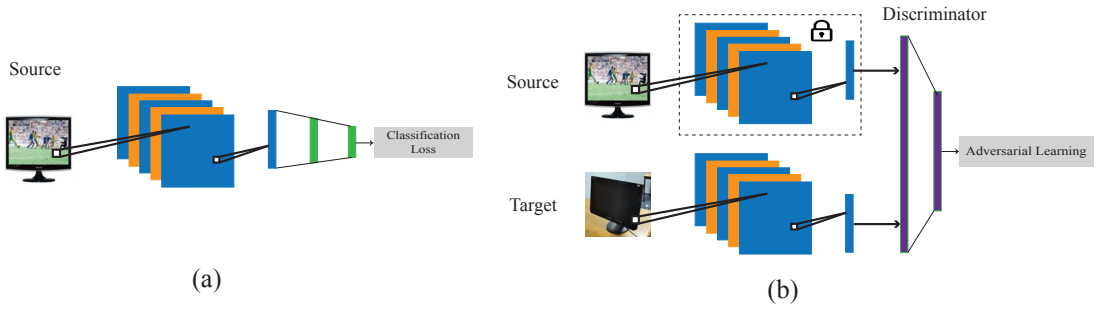

Figure 1.5: ADDA. (a) ADDA first trains a source model using labeled source image examples. (b) It then performs adversarial adaptation by learning a target model such that a discriminator cannot reliably predict source and target samples' domain labels. In testing, target model can be used with source classifier.

$$r_w(\theta_j^s, \theta_j^t) = \exp(||a_j\theta_j^s + b_j - \theta_j^t||_2^2) - 1 \ , \tag{1.8}$$

where θ_j^s and θ_j^t denote the weights of the corresponding parameters of the source and target networks, respectively.

[63], which is a SDA approach, uses two CNN streams, for source and target, fused at the classifier level. It uses the 16-layer VGG model as the base networks, and features from the fully connected layers fc7 of each network are used to compute second or even higher order scatter tensors; one per network stream per class. During adaptation, [63] aligns the scatters of the two network streams of the same class (within-class scatters) while maintaining good separation of the between-class scatters. Our approach presented in Chapter 2 is similar to [63]. Instead of minimizing the distribution distances (second or even higher order scatter tensors in [63]), we minimize the point-wise surrogates of distribution distances for within-class alignment. This is important when we have access to very few labeled target samples in training. For the between-class separation, we do not rely only on the classifier loss but add another loss term using point-wise similarities.

The final goal of these approaches is to maximize the confusion between the source and target samples in the feature space. In other words, they are looking for minimizing the distance between $p(z|x^s)$ and $p(z|x^t)$. So far we have talked about several traditional models to minimize that distance. Recently, adversarial learning [87] has been proposed for image generation. It contains two networks, the generator G and the discriminator D, where the

discriminator tries to distinguish between fake and real images while the generator tries to generate fake images to fool the discriminator. In other words, D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{n \sim p_n(n)}[\log(1 - D(G(n)))] \ , \qquad (1.9)$$

where E denotes statistical expectation, n is a sample from the known distribution $p_n(n)$. D is a discriminator that takes an image and can classify it as real or fake. G is a generator that takes a random noise as input and generates an image from it. D and G can be implemented by convolutional neural networks (CNN).

During the adversarial game, the distance between the probability distribution of the fake images and the real images decreases. Therefore, adversarial learning is becoming attractive for minimizing the KL-divergence between two distributions. We have discussed the use of adversarial learning to find a mapping from the source and target distribution for domain adaptation in the previous section. However, it is obvious from the above discussion that adversarial learning could also be very helpful for domain adaption in finding a shared latent space. For domain adaptation instead, we need to use the adversarial game in the feature space not in the image space. [89] was one of the first papers that used adversarial learning and proposed Adversarial Discriminative Domain Adaptation (ADDA). For classification, ADDA uses three network blocks: The first and second blocks (source and target embeddings) are responsible for embedding images of the source and target domains to their latent representations, respectively, and can be implemented by CNNs. The first and the second blocks are identical. The third block (classifier) is responsible for mapping from the latent space to the label space (doing the classification), and can be implemented by some fully connected layers. ADDA (See Figure 1.5) first learns the source embedding and the classifier using the labels in the source domain in a feedforward fashion. It then initializes the target embedding with the source embedding and uses the basic adversarial learning to map the target images to the source latent space. In other words, it confuses the target latent space with the source latent space. Similar to all other approaches in this category, it is safe to use the classifier trained on the source domain together with the target embedding

after maximum confusion. Since ADDA needs a lot of unlabeled target data for training, it is a UDA method. [88] introduced a coupled generative adversarial network (CoGAN) for learning a joint distribution of multi-domain images for different applications including UDA. [90] introduces an approach that leverages unlabeled data to bring the source and target distributions closer by inducing a symbiotic relationship between the learned embedding and a generative adversarial framework. These methods need a lot of unlabeled target samples in their training. In Chapter 3, we discuss a method for addressing the supervised domain adaptation using adversarial learning, where we need very few labeled target samples in training. Although our method is somewhat similar to ADDA, we provide important contributions to adversarial learning as we discuss in Chapter 3. To the best of our knowledge, there is no previous work which addresses the SDA problem using adversarial learning.

- **Using regularizers.** The third method regularizes a classifier trained on a source distribution to work well on a target distribution [84, 85, 119, 120, 121, 122]. [119] adds a new regularizer into the SVM objective function. It minimizes both the classification error over the training examples, and the discrepancy between the adapted and original classifiers using only the limited target labeled examples. [120] simultaneously learns a kernel function and a robust SVM classifier by minimizing both the structural risk functional of the SVM and the distribution mismatch of labeled and unlabeled samples between the source and target domains. The regularizing method is suitable for SDA and shows a poorer performance compared to shared latent space learning.

In addition to these methods, there are some other methods that do not assume any target data in their training (zero-shot). Instead, these methods use other types of information. For example, [123] uses descriptor of domain and [124] uses task-irrelevant data.

Domain Generalization. Domain generalization (DG) is a less investigated problem and is addressed in two ways. DG assumes that we are given image-label pairs from several similar source distributions. The goal of DG is to learn a mapping from the image space to the label space such that it works well for any unseen distributions. As we show in Chapter 2, simply putting all the image-label pairs coming from several source distributions in one dataset will be suboptimal. DG was first discussed in [54]. [54] develops a distribution-free, kernel-based approach to solve DG by

identifying an appropriate reproducing kernel Hilbert space and optimizing a regularized empirical risk over the space. [125] uses an idea similar to [54] for multiclass domain generalization. [109] learns an invariant transformation by minimizing the dissimilarity across domains, whilst preserving the functional relationship between input and output variables. It shows that reducing dissimilarity improves the expected generalization ability of classifiers on new domains. Our proposed DG follows a similar idea and we try to reduce within-class dissimilarity and also increase between-class dissimilarity.

[126], which can be used for SDA too, finds a representation that minimizes the mismatch between domains and maximizes the separability of data. [127] learns features that are robust to variations across domains.

There are some approaches that exploit all information from the training domains or datasets to train a classifier or regulate its weights [128, 129, 130, 131, 132]. Specifically, [128] adjusts the weights of the classifier to work well on an unseen dataset, and [130] fuses the score of exemplar classifiers given any test sample. While most works use non-deep models, here we approach DG as in the first way, and extend the proposed SDA approach (Chapter 2) by training a deep siamese network to find a shared invariant representation where semantic alignment as well as separation are explicitly accounted for. To the best of our knowledge, [127] is the only DG approach using deep models, and our method is the first deep method that solves both adaptation and generalization.

Learning Using Privileged Information. In some applications main view training data can be paired with additional information which is only available in training. If the additional information is informative about the task at hand, it can help to better train a machine learning model on the main view. [56] was the first work investigating this problem and named this new paradigm as learning using privileged information (LUPI). Privileged information has other names like side information [133] and hidden information [134, 135, 9]. Traditional supervised learning assumes that a training dataset made of N pairs $(x_1, y_1), \cdots, (x_N, y_N)$ is given, where the feature $x_i \in \mathcal{X}$ is a realization from a random variable X, the label $y_i \in \mathcal{Y}$ is a realization from a random variable Y, and the pairs are i.i.d. samples from a joint probability distribution $p(X, Y)$. Under this setting the goal is to learn a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$ by searching over a space of admissible functions $\mathcal{F}$.

The LUPI paradigm assumes that every training data pair comes with *auxiliary* information,

augmenting the training dataset to $(x_1, x_1^*, y_1), \cdots, (x_N, x_N^*, y_N)$. The auxiliary feature $x_i^* \in \mathcal{X}^*$ is a realization from the random variable X^*. The triplets are now i.i.d. samples from the joint distribution $p(X, X^*, Y)$. Under LUPI settings, the goal is to learn a prediction function $f^* : \mathcal{X} \to \mathcal{Y}$ by searching $\mathcal{F}$. Note that in order to predict a label y, at testing time f^* uses only data from the *main* space $\mathcal{X}$. Therefore, the data from the auxiliary space $\mathcal{X}^*$ is only available during training, which is why it is called *privileged*. From the same amount of training samples N, the LUPI classifier f^* will improve the performance of the traditional classifier f [136]. [56] proposed SVM+ to address this problem by modifying the SVM optimization. SVM finds the optimal separating hyperplane, the one that makes a small number of training errors and possesses a large margin. If the target data is separable, the hyperplane can be obtained by minimizing:

$$\begin{aligned} \min_{w,b} \quad & \|w\|^2 \\ \text{s.t. } & y_i(\langle w, x_i \rangle + b) \geq 1 \quad , \forall i \in \{1, \cdots, N\} \ . \end{aligned} \tag{1.10}$$

For the non-separable data one uses non-negative slack variables as follow:

$$\begin{aligned} \min_{w,b} \quad & \|w\|^2 + C \sum_{i=1}^{N} \xi_i \\ \text{s.t. } & y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \ , \ \ \xi_i \geq 0 \ , \ \ \forall i \in \{1, \cdots, N\} \ , \end{aligned} \tag{1.11}$$

where C is the usual parameter to control the slackness. SVM+ [56] uses the privileged data as a proxy for predicting the slack variables by minimizing

$$\begin{aligned} \min_{w,b,w^*,b^*} \quad & \|w\|^2 + \gamma \|w^*\|^2 + C \sum_{i=1}^{N} \xi_i(w^*, b^*) \\ \text{s.t. } & y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i(w^*, b^*) \ , \ \ \xi_i(w^*, b^*) \geq 0 \ , \ \ \forall i \in \{1, \cdots, N\} \ , \\ & \xi_i(w^*, b^*) = \langle w^*, x_i^* \rangle + b^* \ . \end{aligned} \tag{1.12}$$

This approach is equivalent to learning an oracle that tells which sample is easy and which one is hard to predict. There are a lot of works inspired from the oracle teacher idea. It is used in the learning to rank approach introduced in [10], where it is shown that different types of privileged information, such as bounding boxes, attributes, text, and annotator rationales [137] can be used for learning a better classifier for object recognition. [138] also uses a similar framework and

incorporates the LUPI paradigm into mult-view learning, proposing a new SVM-based model. [139] uses the oracle teacher framework and support vector regression for estimating the height. Inspired by SVM+, [140] carefully defines relative attributes for SVM+ to improve the performance of age estimation. In this case, the privileged information enables separation of outliers from inliers at the training stage and effectively manipulates slack variables and age determination errors during model training. [141] uses the oracle teacher framework for privileged multi-label learning.

Privileged information is becoming popular for deep learning [133], specially for action recognition. In action recognition, skeleton data is very informative but hard to obtain in testing. Therefore, there are several works using deep learning with skeleton data as privileged information [142, 143, 144].

In computer vision auxiliary information has been incorporated into the learning process in several forms. For example, in attribute based approaches [145, 146] labeled data is used for training extra attribute classifiers to extract mid-level features. Similarly, [147, 148] learn to extract mid-level features by training data from additionally annotated concepts. Our framework differs from those because the auxiliary information can be generic, and because it is used for improving the classifier performance in a single optimization framework, whereas attribute classifiers may be disconnected from the main classification task. Another form of auxiliary information is given by the structure of the hidden domain of latent models for object detection and gesture recognition [149, 150]. Compared to those approaches we use information from auxiliary labeled data.

Our approach exploits the privileged information differently. An information theoretic framework learns how to compress the source domain for doing prediction in a way that is as informative of the privileged source domain as possible, regardless of the type of classifier used, and without tying privileged information to slack variables. This is done by extending the original IB method [8], often used for clustering [151], and also used in [152] for incorporating "negative information" that is irrelevant for the task at hand, and that should not be learned by the representation. This is similar to [153], where negative information is used for face recognition with discounted pose-induced similarity.

The LUPI paradigm has recently been used for boosting [154], for object localization in a structured prediction framework [155], for facial feature detection [10], for metric learning [156, 157], in a logistic classification framework [134], in a max-margin latent variable model [135], in

matrix completion [158], in person re-identification [159], and in active learning [160]. In the above methods, either the problem settings or the approaches taken are significantly different from the information theoretic principles that are driving our program. Other recent approaches include [161, 162], which focus on the missing view problem by discriminatively learning projections to a shared latent subspace. This approach relates more to multi-view learning, but only the main view pipeline is used for testing, without considering the intrinsic asymmetry of the LUPI framework, as pointed out in [9]. There they propose two principles to learn with auxiliary information based on looking at it as additional features, or as additional labels, where they make assumptions on its informative content. We also introduce a new principle that shares the benefits of their framework, but by using an information theoretic approach we have no need to make distinctions between the types of auxiliary information, and we have no need to state requirements on the information content.

Domain Adaptation with Privileged Information. This problem has received very limited attention in the literature. The first work addressing this problem is [59, 163], and the extension for web data [94]. They jointly learn a multi-class large-margin classifier, as well as two projections for the main and auxiliary views, respectively. This is done while maximizing the correlation among views, as well as minimizing the distribution mismatch according to the MMD. [164] uses the idea of oracle teacher to learn an adaptive SVM+, combining the advantages of both the LUPI paradigm and the domain adaptation framework. [144] leverages both a large-scale dataset and its extra modalities, to learn a better model for temporal action detection and action classification without needing to have access to these modalities during test time. On the other hand, we extend the information bottleneck method into a general principle that handles the auxiliary view as well as the distribution mismatch from a single information theoretic point of view in Chapter 5. Computationally, this entails the estimation of only one projection, rather than two. It allows handling source data points with missing auxiliary view, and works with any classifiers (shallow and deep ones).

Chapter 2

Few-Shot Domain Adaptation and Generalization

2.1 Introduction

This chapter provides a unified framework for addressing the problem of visual supervised domain adaptation (SDA) and domain generalization (DG) with deep models. It can work with any network structure, and the SDA approach requires very few labeled target samples per category in training. We may use "few-shot domain adaptation" instead of the term "supervised domain adaptation" in this Chapter to stress the fact that we have very few target samples per class in training. The main idea is to exploit the Siamese architecture to learn an embedding subspace that is discriminative, and where mapped visual domains are semantically aligned and yet maximally separated. The few-shot setting becomes attractive especially when only few target data samples need to be labeled. In this scenario, alignment and separation of semantic probability distributions is difficult because of the lack of data. We found that by reverting to point-wise surrogates of distribution distances and similarities provides an effective solution. In addition, the approach has a high speed of adaptation, which re- quires an extremely low number of labeled target training samples, even one per category can be effective. Moreover, the approach is also robust to adapting to categories that have no target labeled samples. Although domain adaptation and generalization are closely related, adaptation techniques are not directly applied to DG, and vice versa. However, we show that by making simple changes to our proposed train- ing loss function, and by maintaining the same architecture, our SDA approach very effectively extends to DG.

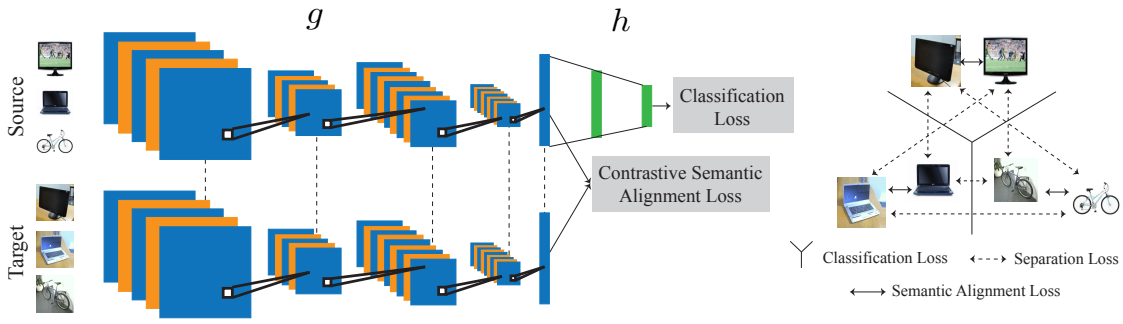

Figure 2.1: Deep Few-Shot domain adaptation. In training, the semantic alignment loss minimizes the distance between samples from different domains but same class label and the separation loss maximizes the distance between samples from different domains and class labels. At the same time, the classification loss guarantees high classification accuracy.

Using basic principles, we analyze how visual classification is extended to handle UDA by aligning a source do- main distribution to a target domain distribution to make the classifier domain invariant. This leads to observing that SDA approaches improve upon UDA by making the alignment semantic, because they can ensure the alignment of semantically equivalent distributions from different domains. However, we go one step ahead by suggesting that semantic distribution separation should further increase performance, and this leads to the introduction of a classification and contrastive semantic alignment (CCSA) loss. We deal with the limited size of target domain samples by observing that the CCSA loss relies on computing distances and similarities between distributions (as typically done in adaptation and generalization approaches). Those are difficult to represent with limited data. Thus, we revert to point-wise surrogates. The resulting approach turns out to be very effective as shown in the experimental section. For both SDA and DG, the experiments show very promising results.

2.2 Few-Shot DA with Scarce Target Data

In this section we describe the model we propose to address supervised domain adaptation (SDA), and in the Section 2.4 we extend it to address the domain generalization problem. We are given a training dataset made of pairs $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^N$. The feature $x_i^s \in \mathcal{X}$ is a realization from a random variable X^s, and the label $y_i^s \in \mathcal{Y}$ is a realization from a random variable Y. In addition, we are also given the training data $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^M$, where $x_i^t \in \mathcal{X}$ is a realization from a random variable X^t, and the labels $y_i^t \in \mathcal{Y}$. We assume that there is a *covariate shift* [52] between X^s and

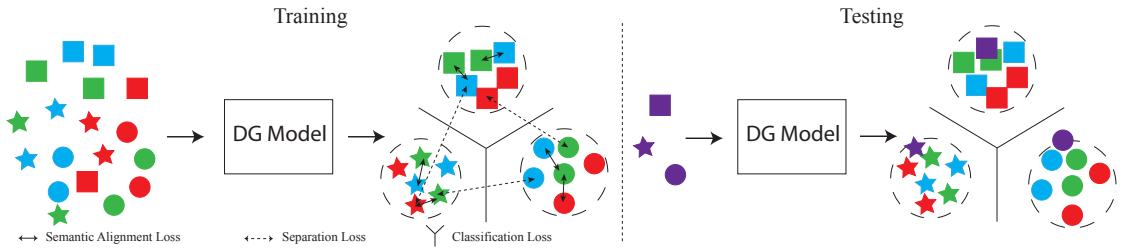

Figure 2.2: Deep domain generalization. In training, the semantic alignment loss minimizes the distance between samples from different domains but the same class label and the separation loss maximizes the distance between samples from different domains and class labels. At the same time, the classification loss guarantees high classification accuracy. In testing, the embedding function embeds samples from unseen distributions to the domain invariant space and the prediction function classifies them (right). In this figure, different colors represent different domain distributions and different shapes represent different classes.

X^t, i.e., there is a difference between the probability distributions $p(X^s)$ and $p(X^t)$. We say that X^s represents the *source domain* and that X^t represents the *target domain*. Under this settings the goal is to learn a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that during testing is going to perform well on data from the target domain.

The problem formulated thus far is typically referred to as *supervised domain adaptation*. In this work we are especially concerned with the version of this problem where only very few target labeled samples per class are available. We aim at handling cases where there is only one target labeled sample, and there can even be some classes with no target samples at all.

2.2.1 Deep SDA

In the absence of covariate shift a visual classifier f is trained by minimizing a *classification loss*

$$\mathcal{L}_C(f) = E[\ell(f(X^s), Y)] \,, \tag{2.1}$$

where $E[\cdot]$ denotes statistical expectation and ℓ could be any appropriate loss function (for example categorical cross-entropy for multi-class classification). When the distributions of X^s and X^t are different, a deep model f_s trained with $\mathcal{D}_s$ will have reduced performance on the target domain. Increasing it would be trivial by simply training a new model f_t with data $\mathcal{D}_t$. However, $\mathcal{D}_t$ is small and deep models require large amounts of labeled data.

In general, f could be modeled by the composition of two functions, i.e., $f = h \circ g$. Here $g : \mathcal{X} \rightarrow \mathcal{Z}$ would be an embedding from the input space $\mathcal{X}$ to a feature or embedding space $\mathcal{Z}$,

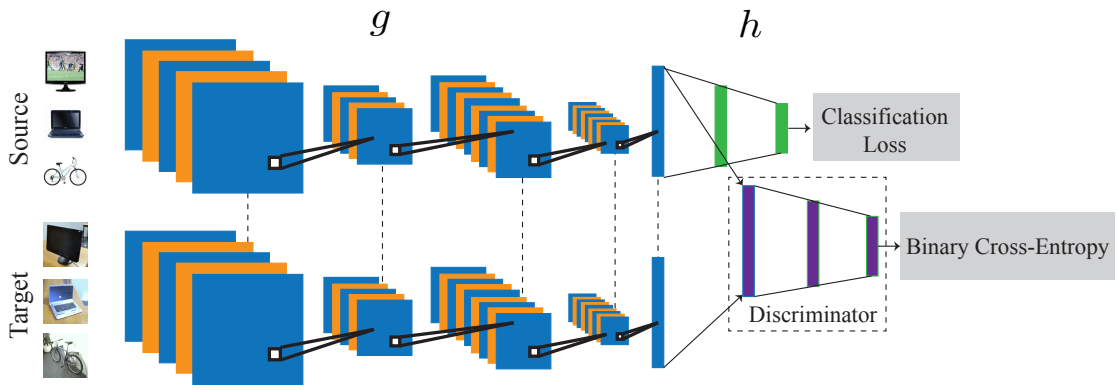

Figure 2.3: CCSA with the discriminator choice. The pairs in the embedding space are concatenated and passed to the discriminator.

and $h : \mathcal{Z} \to \mathcal{Y}$ would be a function for predicting from the feature space. With this notation we would have $f_s = h_s \circ g_s$ and $f_t = h_t \circ g_t$, and the SDA problem would be about finding the best approximation for g_t and h_t, given the constraints on the available data.

The unsupervised DA paradigm (UDA) assumes that $\mathcal{D}_t$ does not have labels. In that case the typical approach assumes that $g_t = g_s = g$, and f minimizes (2.1), while g also minimizes

$$\mathcal{L}_{CA}(g) = d(p(g(X^s)), p(g(X^t))) \; . \tag{2.2}$$

The purpose of (2.2) is to align the distributions of the features in the embedding space, mapped from the source and the target domains. d is meant to be a metric between distributions that once aligned, they will no longer allow to tell whether a feature is coming from the source or the target domain. For that reason, we refer to (2.2) as the *confusion alignment* loss. A popular choice for d is the Maximum Mean Discrepancy [3]. In the embedding space $\mathcal{Z}$, features are assumed to be domain invariant. Therefore, UDA methods say that from the feature to the label space it is safe to assume that $h_t = h_s = h$.

Since we are interested in visual recognition, the embedding function g would be modeled by a convolutional neural network (CNN) with some initial convolutional layers, followed by some fully connected layers. In addition, the training architecture would have two streams, one for source and the other for target samples. Since $g_s = g_t = g$, the CNN parameters would be shared as in a Siamese architecture. In addition, the source stream would continue with additional fully connected layers for modeling h. See Figure 2.1.

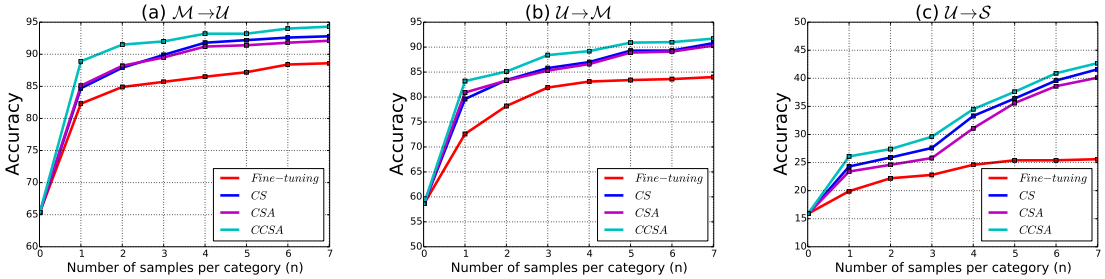

Figure 2.4: Average classification accuracy for the $\mathcal{M} \rightarrow \mathcal{U}$, $\mathcal{U} \rightarrow \mathcal{M}$, and $\mathcal{U} \rightarrow \mathcal{S}$ tasks for different number of labeled target samples per category (n). It shows that our model provides significant improvement over baselines.

From the above discussion it is clear that in order to perform well, UDA needs to align effectively. This can happen only if distributions are represented by a sufficiently large dataset. Therefore, UDA approaches are in a position of weakness because we assume $\mathcal{D}_t$ to be small. Moreover, UDA approaches have also another intrinsic limitation, which is that even with perfect confusion alignment, there is no guarantee that samples from different domains but the same class label, would map nearby in the embedding space. This lack of *semantic alignment* is a major source of performance reduction.

SDA approaches easily address the semantic alignment problem by replacing (2.2) with

$$\mathcal{L}_{SA}(g) = \sum_{a=1}^{C} d(p(g(X_a^s)), p(g(X_a^t))) \ , \tag{2.3}$$

where C is the number of class labels, and $X_a^s = X^s|\{Y = a\}$ and $X_a^t = X^t|\{Y = a\}$ are conditional random variables. d instead is a suitable distance mesure between the distributions of X_a^s and X_a^t in the embedding space. We refer to (2.3) as the *semantic alignment* loss, which clearly encourages samples from different domains but the same label, to map nearby in the embedding space.

While the analysis above clearly indicates why SDA provides superior performance than UDA, it also suggests that deep SDA approaches have not considered that greater performance could be achieved by encouraging class separation, meaning that samples from different domains and with different labels, should be mapped as far apart as possible in the embedding space. This idea means that, in principle, a semantic alignment less prone to errors should be achieved by adding to (2.3)

the following term

$$\mathcal{L}_S(g) = \sum_{a,b|a\neq b} k(p(g(X_a^s)), p(g(X_b^t))) \ , \tag{2.4}$$

where k is a suitable similarity mesure between the distributions of X_a^s and X_b^t in the embedding space, which adds a penalty when the distributions $p(g(X_a^s))$ and $p(g(X_b^t))$ come close, since they would lead to lower classification accuracy. We refer to (2.4) as the *separation* loss.

Finally, we suggest that SDA could be approached by learning a deep model $f = h \circ g$ such that

$$\mathcal{L}_{CCSA}(f) = \mathcal{L}_C(h \circ g) + \mathcal{L}_{SA}(g) + \mathcal{L}_S(g) \ . \tag{2.5}$$

We refer to (2.5) as the *classification and contrastive semantic alignment* loss. This would allow to set $g_s = g_t = g$. The classification network h is trained only with source data, so $h_s = h$. In addition, to improve performance on the target domain, h_t could be obtained via fine-tuning based on the few samples in $\mathcal{D}_t$, i.e.,

$$h_t = \text{fine-tuning}(h|\mathcal{D}_t) \ . \tag{2.6}$$

Note that the network architecture remains the one in Figure 2.1, only with a different loss, and training procedure.

2.2.2 Handling Scarce Target Data

When the size of the labeled target training dataset $\mathcal{D}_t$ is very small, minimizing the loss (2.5) becomes a challenge. The problem is that the semantic alignment loss as well as the separation loss rely on computing distances and similarities between distributions, and those are very difficult to represent with as few as one data sample.

Rather than attempting to characterize distributions with statistics that require enough data, because of the reduced size of $\mathcal{D}_t$, we compute the distance in the semantic alignment loss (2.3) by computing average pairwise distances between points in the embedding space, i.e., we compute

$$d(p(g(X_a^s)), p(g(X_a^t))) = \sum_{i,j} d(g(x_i^s), g(x_j^t)) \ , \tag{2.7}$$

where it is assumed $y_i^s = y_j^t = a$. The strength of this approach is that it allows even a single labeled target sample to be paired with all the source samples, effectively trying to semantically

align the entire source data with the few target data. Similarly, we compute the similarities in the separation loss (2.4) by computing average pairwise similarities between points in the embedding space, i.e., we compute

$$k(p(g(X_a^s)), p(g(X_b^t))) = \sum_{i,j} k(g(x_i^s), g(x_j^t)) \ , \tag{2.8}$$

where it is assumed that $y_i^s = a \neq y_j^t = b$.

2.3 Distance and Similarity Choices

The performance of our model also depends on the types of distances (2.7) and similarities (2.8). We define three approaches and discussed which one may provide a better performance. For all the choices, to balance the classification versus the contrastive semantic alignment portion of the loss (2.5), (2.7) and (2.8) are normalized and weighted by $1 - \gamma$ and (2.1) by γ.

2.3.1 First Choice - Discriminator

In the previous section we showed that creating pairs between source and target samples can be effective for scarce target data. In other words, we can create positive pairs (when images of pairs come from the same class label) and negative pairs (when images of pairs come from different class labels) and minimize their distances and their similarities in the embedding space, respectively. Inspired by recent works of the image generation task [87], we proposed to use a discriminator to compute distances and similarities. The discriminator consists of few fully connected layers and one output with sigmoid activation. This is important because the discriminator would give similarity scores to input pairs from 0 to 1 (0 as least similarity and 1 as least distance). If input pairs come from the same class, the output of the discriminator would ideally be 1. Similarly, if they come from different classes, the output would ideally be 0. Therefore, a similarity score can be converted to a distance score by

$$d(p(g(X_a^s)), p(g(X_a^t))) = 1 - k(p(g(X_a^s)), p(g(X_a^t))) \ , \tag{2.9}$$

for positive pairs.

During the implementation, the pairs in the embedding space are concatenated and passed to the discriminator as shown in the Figure 2.3. Since the distribution of the data in the embedding space is unknown, we cannot find a good distance function to measure the distances in that space (L2 norm may not be a good option). Therefore, instead of finding a distance metric, we let the discriminator computes the distances/similarities.

2.3.2 Second Choice - Contrastive Loss

Since we are dealing with scarce target data, we want to keep the number of parameters low (more parameters likely needs more data to be trained). Using a discriminator to compute distances will increase the number of parameters of our model which may leads to performance reduction. Discriminator may also suffer from the labeling of training pairs. We did not differentiate between negative pairs as we label all of them 1. This also may cause performance reduction because some negative pairs may require more attention (giving more penalty to the network).

To address the above limitations, we assume that

$$d(g(x_i^s), g(x_j^t)) = \frac{1}{2}\|g(x_i^s) - g(x_j^t)\|^2 , \tag{2.10}$$

$$k(g(x_i^s), g(x_j^t)) = \frac{1}{2}\max(0, m - \|g(x_i^s) - g(x_j^t)\|)^2 \tag{2.11}$$

where $\|\cdot\|$ denotes the Frobenius norm, and m is the margin that specifies the separability in the embedding space. Note that with the choices outlined in (2.10) and (2.11), the loss $\mathcal{L}_{SA}(g)+\mathcal{L}_S(g)$ becomes the well known Contrastive loss as defined in [1].

These choices do not increase the number of parameters of the model, and will give more penalty to those negative pairs that have small distances (ideally we want negative pairs to be as far as possible). However, this method may suffer from the Euclidean distance metric.

2.3.3 Third Choice - Triplet Loss

Distances (2.7) and similarities (2.8) of our model is also can be seen as Triplet loss [2]. In this scenario, we assume each target sample as an anchor (x_i^t) paired with one positive source sample ($g(x_i^{s^p})$, coming from the same class as an anchor) and one negative source sample ($g(x_i^{s^n})$, coming from different class). With this pairing method, the loss $\mathcal{L}_{SA}(g)+\mathcal{L}_S(g)$ becomes the Triplet loss [2]

Table 2.1: Digits datasets. Classification accuracy for domain adaptation over the MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), and SVHN datasets $\mathcal{S}$. LB is our base model without adaptation. CCSA - n stands for our method when we use n labeled target samples per category in training.

	LB	[89]	[88]	[90]	SDA Method	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
$\mathcal{M} \to \mathcal{U}$	65.4	89.4	91.2	92.5	Fine-tuning	82.3	84.9	85.7	86.5	87.2	88.4	88.6
					SADA [78]	89.1	91.3	91.9	93.3	93.4	94.0	94.4
					CCSA-First	85.0	89.4	90.5	91.9	91.9	93.4	93.4
					CCSA-Second	88.9	91.5	92.0	93.2	93.2	94.0	94.3
					CCSA-Third	89.2	91.6	92.2	93.2	93.1	94.5	94.7
$\mathcal{U} \to \mathcal{M}$	58.6	90.1	89.1	90.8	Fine-tuning	72.6	78.2	81.9	83.1	83.4	83.6	84.0
					SADA [78]	81.1	84.2	87.5	89.9	91.1	91.2	91.5
					CCSA-First	78.2	81.6	85.7	87.3	88.3	89.1	90.1
					CCSA-Second	83.2	85.1	88.4	89.2	90.9	91.0	91.7
					CCSA-Third	81.0	84.3	88.0	89.1	90.6	90.8	91.8
$\mathcal{M} \to \mathcal{S}$	25.3	-	-	36.4	Fine-tuning	29.7	31.2	36.1	36.7	38.1	38.3	39.1
					SADA [78]	37.7	40.5	42.9	46.3	46.1	46.8	47.0
					CCSA-First	18.1	25.1	29.9	30.5	34.8	38.6	39.5
					CCSA-Second	20.3	25.9	30.6	31.6	34.6	37.5	40.2
					CCSA-Third	20.6	25.6	32.0	33.9	35.7	39.5	41.2
$\mathcal{U} \to \mathcal{S}$	15.9	-	-	-	Fine-tuning	19.9	22.2	22.8	24.6	25.4	25.4	25.6
					SADA [78]	27.5	29.8	34.5	36.0	37.9	41.3	42.9
					CCSA-First	23.6	25.6	28.4	30.8	35.9	38.9	38.4
					CCSA-Second	26.1	27.4	29.6	34.5	37.6	40.9	42.7
					CCSA-Third	25.1	28.1	28.6	35.6	36.1	39.1	41.2

$$\sum_{i}^{N} [d(g(x_i^t), g(x_i^{s^p})) - d(g(x_i^t), g(x_i^{s^n})) + \alpha] \,, \tag{2.12}$$

where α is a margin that is enforced between positive and negative pairs, N is the number of all possible triplets, and d is Frobenius norm,. We used hard triplets selection [2] in our implementation.

2.4 Extension to Domain Generalization

In visual domain generalization (DG), D labeled datasets $\mathcal{D}_{s_1}$, $\cdots$, $\mathcal{D}_{s_D}$, representative of D distinct source domains are given. The goal is to learn from them a visual classifier f that during testing is going to perform well on data $\mathcal{D}_t$, not available during training, thus representative of an unknown target domain.

The SDA method in Section 2.2 treats source and target datasets $\mathcal{D}_s$ and $\mathcal{D}_t$ almost symmetrically. In particular, the embedding g aims at achieving semantic alignment, while favoring class separation. The only asymmetry is in the prediction function h that is trained only on the source, to be then fine-tuned on the target.

In domain generalization, we are not interested in adapting the classifier to the target domain, because it is unknown. Instead, we want to make sure that the embedding g maps to a domain

Table 2.2: Office dataset. Classification accuracy for domain adaptation over the 31 categories of the Office dataset. $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$ stand for Amazon, Webcam, and DSLR domain. `Lower Bound` is our base model without adaptation.

		Unsupervised			Supervised			CCSA		
	Lower Bound	[165]	[65]	[64]	[62]	[63]	[78]	First	Second	Third
$\mathcal{A} \to \mathcal{W}$	61.2 ± 0.9	61.8 ± 0.4	68.5 ± 0.4	68.7 ± 0.3	82.7 ± 0.8	84.5 ± 1.7	88.1 ± 1.2	85.3 ± 1.1	**88.1 ± 1.0**	87.6 ± 1.2
$\mathcal{A} \to \mathcal{D}$	62.3 ± 0.8	64.4 ± 0.3	67.0 ± 0.4	67.1 ± 0.3	86.1 ± 1.2	86.3 ± 0.8	88.2 ± 1.0	85.3 ± 1.5	**89.2 ± 1.2**	88.5 ± 1.1
$\mathcal{W} \to \mathcal{A}$	51.6 ± 0.9	52.2 ± 0.4	53.1 ± 0.3	54.09 ± 0.5	65.0 ± 0.5	65.7 ± 1.7	71.1 ± 0.9	67.3 ± 1.2	72.0 ± 1.5	**72.1 ± 1.0**
$\mathcal{W} \to \mathcal{D}$	95.6 ± 0.7	98.5 ± 0.4	**99.0 ± 0.2**	**99.0 ± 0.2**	97.6 ± 0.2	97.5 ± 0.7	97.5 ± 0.6	97.0 ± 0.8	97.6 ± 0.4	98.3 ± 0.2
$\mathcal{D} \to \mathcal{A}$	58.5 ± 0.8	52.1 ± 0.8	54.0 ± 0.4	56.0 ± 0.5	66.2 ± 0.3	66.5 ± 1.0	68.1 ± 0.6	69.2 ± 0.5	**71.9 ± 0.4**	71.7 ± 0.5
$\mathcal{D} \to \mathcal{W}$	80.1 ± 0.6	95.0 ± 0.5	96.0 ± 0.3	**96.4 ± 0.3**	95.7 ± 0.5	95.5 ± 0.6	96.4 ± 0.8	95.8 ± 0.9	**96.4 ± 0.8**	**96.4 ± 0.8**
Average	68.2	70.6	72.9	73.6	82.2	82.6	84.9	83.3	**85.8**	85.7

invariant space. To do so we consider every distinct unordered pair of source domains (u, v), represented by $\mathcal{D}_{s_u}$ and $\mathcal{D}_{s_v}$, and, like in SDA, impose the semantic alignment loss (2.3) as well as the separation loss (2.4). Moreover, the losses are summed over every pair in order to make the map g as domain invariant as possible. Similarly, the classifier h should be as accurate as possible for any of the mapped samples, to maximize performance on an unseen target. This calls for having a fully symmetric learning for h by training it on all the source domains, meaning that the classification loss (2.1) is summed over every domain s_u. See Figure 2.2.

The network architecture is still the one in Figure 2.1, and we have implemented it with the same choices for distances and similarities as those made in Section 2.2.2 as follow: Given D labeled datasets $\mathcal{D}_{s_1}, \cdots, \mathcal{D}_{s_D}$, representative of D distinct source domains, the end-to-end learning of $f = h \circ g$ is done by minimizing the loss function

$$\begin{aligned} \mathcal{L}_{CCSA}(f) &= \frac{1-\gamma}{D} \sum_u \mathcal{L}_C(h \circ g | \mathcal{D}_{s_u}) \\ &+ \frac{2\gamma}{D^2 - D} \left(\sum_{(u,v)} \mathcal{L}_{SA}(g|\mathcal{D}_{s_u}, \mathcal{D}_{s_v}) + \sum_{(u,v)} \mathcal{L}_S(g|\mathcal{D}_{s_u}, \mathcal{D}_{s_v}) \right) \end{aligned} \tag{2.13}$$

During training, because every unordered pair of domains (u, v) is considered, we control the number of positive and negative training pairs by randomly picking a fraction of all the possible training pairs.

2.5 Experiments

We divide the experiments into two parts, domain adaptation and domain generalization. In both sections, we use benchmark datasets and compare our domain adaptation model and our

Table 2.3: Office dataset. Classification accuracy for domain adaptation over the Office dataset when only the labeled target samples of 15 classes are available during training. Testing is done on all 31 classes. $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$ stand for Amazon, Webcam, and DSLR domain. `Lower Bound` is our base model without adaptation.

			CCSA		
	Lower Bound	[62]	First	Second	Third
$\mathcal{A} \to \mathcal{W}$	52.1 ± 0.6	59.3 ± 0.6	61.5 ± 1.2	**63.9 ± 0.8**	63.1 ± 0.9
$\mathcal{A} \to \mathcal{D}$	61.6 ± 0.8	68.0 ± 0.5	66.1 ± 0.7	**70.8 ± 0.5**	**70.8 ± 0.9**
$\mathcal{W} \to \mathcal{A}$	34.5 ± 0.9	40.5 ± 0.2	38.8 ± 1.2	**43.9 ± 0.9**	43.6 ± 1.0
$\mathcal{W} \to \mathcal{D}$	95.1 ± 0.2	**97.5 ± 0.1**	95.5 ± 0.4	97.2 ± 0.3	**97.5 ± 0.1**
$\mathcal{D} \to \mathcal{A}$	40.1 ± 0.3	**43.1 ± 0.2**	41.9 ± 0.6	**43.1 ± 0.2**	42.6 ± 0.6
$\mathcal{D} \to \mathcal{W}$	89.7 ± 0.8	90.0 ± 0.2	90.0 ± 0.2	91.1 ± 0.2	**91.6 ± 0.3**
Average	62.26	66.4	65.6	**68.3**	68.2

Table 2.4: Office dataset. Classification accuracy for domain adaptation over the 10 categories of the Office dataset. $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$ stand for Amazon, Webcam, and DSLR domain. `Lower Bound` is our base model with no adaptation.

						CCSA		
	Lower Bound	GFK [66]	mSDA [166]	CDML [167]	RTML [168]	First	Second	Third
	SURF							
$\mathcal{A} \to \mathcal{W}$	26.5 ± 3.1	39.9 ± 0.9	35.5 ± 0.5	37.3 ± 0.7	43.4 ± 0.9	68.3 ± 1.3	71.5 ± 1.1	**72.8 ± 1.0**
$\mathcal{A} \to \mathcal{D}$	17.5 ± 1.2	36.2 ± 0.7	29.7 ± 0.7	35.3 ± 0.5	43.3 ± 0.6	73.6 ± 1.5	74.5 ± 1.1	**75.1 ± 0.9**
$\mathcal{W} \to \mathcal{A}$	25.9 ± 1.0	29.8 ± 0.6	32.1 ± 0.8	32.4 ± 0.5	37.5 ± 0.7	41.8 ± 0.8	**43.9 ± 0.7**	43.2 ± 0.9
$\mathcal{W} \to \mathcal{D}$	46.9 ± 1.1	80.9 ± 0.4	56.6 ± 0.4	77.9 ± 0.9	**91.7 ± 1.1**	84.6 ± 1.2	86.2 ± 1.1	88.6 ± 0.9
$\mathcal{D} \to \mathcal{A}$	19.3 ± 1.9	33.2 ± 0.6	33.6 ± 0.8	29.4 ± 0.8	**36.3 ± 0.3**	27.6 ± 0.4	31.2 ± 1.1	30.5 ± 0.9
$\mathcal{D} \to \mathcal{W}$	48.0 ± 2.1	79.4 ± 0.6	68.6 ± 0.7	79.4 ± 0.6	**90.5 ± 0.7**	76.5 ± 0.8	78.1 ± 0.9	79.0 ± 0.8
Average	30.6	43.5	38.4	43.5	49.8	62.0	64.2	**64.8**
	DeCaF-fc6							
$\mathcal{A} \to \mathcal{W}$	78.9 ± 1.8	73.1 ± 2.8	64.6 ± 4.2	75.9 ± 2.1	79.5 ± 2.6	93.1 ± 1.8	94.9 ± 1.8	**95.5 ± 1.7**
$\mathcal{A} \to \mathcal{D}$	79.2 ± 2.1	82.6 ± 2.1	72.6 ± 3.5	81.4 ± 2.6	83.8 ± 1.7	96.5 ± 1.5	**97.2 ± 1.0**	**97.2 ± 1.0**
$\mathcal{W} \to \mathcal{A}$	77.3 ± 1.1	82.6 ± 1.3	71.4 ± 1.7	86.3 ± 1.6	90.8 ± 1.6	91.2 ± 0.8	**91.5 ± 0.9**	91.2 ± 0.8
$\mathcal{W} \to \mathcal{D}$	96.6 ± 1.0	98.8 ± 0.9	99.5 ± 0.6	99.4 ± 0.4	**100 ± 0.0**	99.4 ± 0.4	99.6 ± 0.4	99.6 ± 0.4
$\mathcal{D} \to \mathcal{A}$	84.0 ± 1.3	85.4 ± 0.7	78.8 ± 0.5	88.4 ± 0.5	90.6 ± 0.5	91.4 ± 0.8	92.1 ± 0.8	**92.6 ± 0.7**
$\mathcal{D} \to \mathcal{W}$	96.7 ± 0.9	91.3 ± 0.4	97.5 ± 0.4	95.1 ± 0.5	98.6 ± 0.3	98.4 ± 0.4	**98.7 ± 0.6**	**98.7 ± 0.6**
Average	85.4	85.63	80.73	87.75	90.55	95.0	95.6	**95.8**

domain generalization model, both indicated as CCSA, with the state-of-the-art.

2.5.1 Domain Adaptation

We present results using the Office dataset [13], the MNIST dataset [6], the USPS dataset [169], and the SVHN dataset [7].

Digits Datasets

The MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), and SVHN ($\mathcal{S}$) datasets have recently become popular for domain adaptation [170, 118, 89]. They contain images of digits from 0 to 9. We considered six cross-domain tasks. The first two tasks include $\mathcal{M} \to \mathcal{U}, \mathcal{U} \to \mathcal{M}$, and followed the experimental setting

Table 2.5: VLCS dataset. Classification accuracy for domain generalization over the 5 categories of the VLCS dataset. LB (Lower Bound) is our base model trained without the contrastive semantic alignment loss. 1NN stands for first nearest neighbor.

							CCSA	
	Lower Bound			Domain Generalization				
	1NN	SVM	LB	UML [129]	LRE-SVM [130]	SCA [126]	First	Second
$\mathcal{L}, \mathcal{C}, \mathcal{S} \rightarrow \mathcal{V}$	57.2	58.4	59.1	56.2	60.5	64.3	65.2	**67.3**
$\mathcal{V}, \mathcal{C}, \mathcal{S} \rightarrow \mathcal{L}$	52.4	55.2	55.6	58.5	59.7	59.6	60.6	**62.4**
$\mathcal{V}, \mathcal{L}, \mathcal{S} \rightarrow \mathcal{C}$	90.5	85.1	86.1	91.1	88.1	88.9	90.3	**92.3**
$\mathcal{V}, \mathcal{L}, \mathcal{C} \rightarrow \mathcal{S}$	56.9	55.2	54.6	58.4	54.8	59.2	58.5	**59.3**
$\mathcal{C}, \mathcal{S} \rightarrow \mathcal{V}, \mathcal{L}$	55.0	55.5	55.3	56.4	55.0	**59.5**	59.3	**59.5**
$\mathcal{C}, \mathcal{L} \rightarrow \mathcal{V}, \mathcal{S}$	52.6	51.8	50.9	**57.4**	52.8	55.9	56.1	56.9
$\mathcal{V}, \mathcal{C} \rightarrow \mathcal{L}, \mathcal{S}$	56.6	59.9	60.1	55.4	58.8	**60.7**	59.5	60.5
Average	60.1	60.1	60.2	61.5	61.4	64.0	64.2	**65.4**

in [170, 118, 88, 89, 90], which involves randomly selecting 2000 images from MNIST and 1800 images from USPS. To be consistent with literature, we used all training samples of the source domain for training and all testing samples of the target domain for testing for the rest of the cross-domain tasks ($\mathcal{M} \rightarrow \mathcal{S}$, $\mathcal{S} \rightarrow \mathcal{M}$, $\mathcal{U} \rightarrow \mathcal{S}$, and $\mathcal{S} \rightarrow \mathcal{U}$).

Here, we randomly selected n labeled samples per class from target domain data and used them in training. We evaluated our approach for n ranging from 1 to 7 and repeated each experiment 10 times (we only show the mean of the accuracies because the standard deviation is very small).

Similar to [6], we used 2 convolutional layers with 6 and 16 filters of 5×5 kernels followed by max-pooling layers and 2 fully connected layers with size 120 and 84 as the embedding function g, and one fully connected layer with softmax activation as the prediction function h. We compare our method with 3 recent UDA methods based on adversarial learning. Those methods use all target samples in their training stage, while we only use very few labeled target samples per category in training. We also compare our model with the recent few-shot method based on adversarial learning [78]. Table 2.1 shows the average classification accuracy of the digits datasets. CCSA - first, CCSA - second, and CCSA - third correspond to our model with discriminator, Contrastive, and Triplet choices, respectively. CCSA for all choices works well compare to simple fin-tuning. Table 2.1 shows that our few-shot formulation works well even when only one target sample per category ($n = 1$) is available in training. Also, we can see that by increasing n, the accuracy quickly converges to the top. As shown in the table, CCSA - third works better compared to the other two choices.

Ablation study. Separation loss is one of the advantages of the proposed model. We consider three baselines to compare with it. for the $\mathcal{M} \rightarrow \mathcal{U}$ task. First, we train the network with source data and then fine-tune it with available target data. Second, we train the network using the classification and semantic alignment losses ($\mathcal{L}_{CSA}(f) = \mathcal{L}_C(h \circ g) + \mathcal{L}_{SA}(g)$). Third, we train the network using the classification and separation losses ($\mathcal{L}_{CS}(f) = \mathcal{L}_C(h \circ g) + \mathcal{L}_S(g)$). Figures 2.4 show the average accuracies over 10 repetition for `CCSA-Second` for 3 tasks. It show that CSA and CS improve the accuracy over fine-tuning. It is important to see CS in some cases outperforms CSA when there is $n = 1$ and $n = 2$. This shows the importance of the separation loss. Figures 2.4 also show that using the proposed `CCSA` loss (using both semantic alignment and separation losses) provides the best performance. This pattern is seen for all other cross domain tasks.

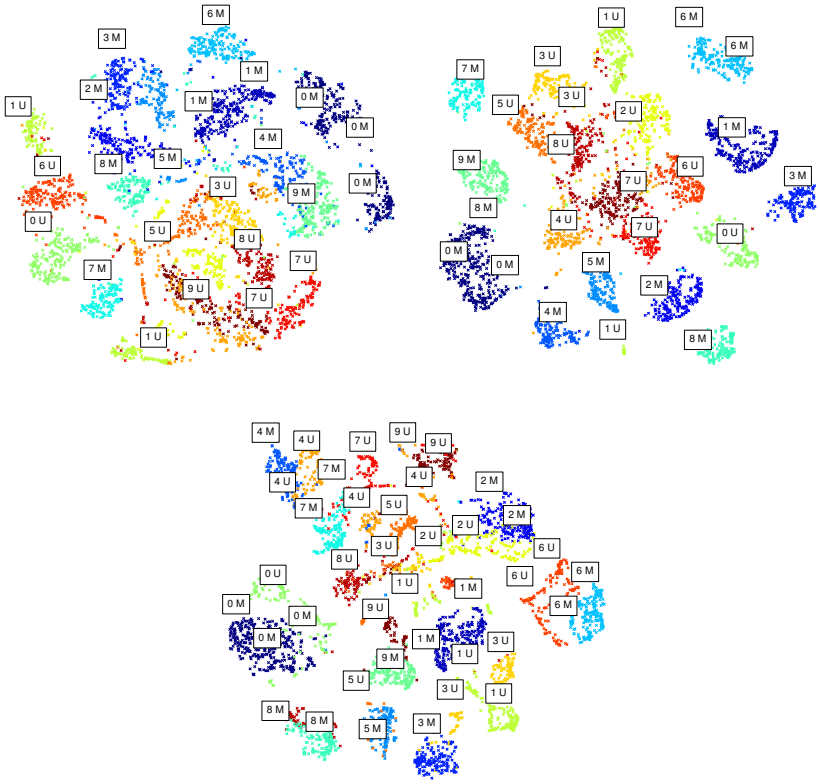

Figure 2.5: Visualization of the MNIST-USPS datasets. Left: 2D visualization of the row images of the MNIST-USPS datasets. The samples from the same class and different domains lie far from each other on the 2D subspace. Middle: 2D visualization of the embedded images using our base model (without domain adaptation). The samples from the same class and different domains still lie far from each other on the 2D subspace. Right: 2D visualization of the embedded images using our SDA model. The samples from the same class and different domains lie very close to each other on the 2D subspace.

Visualization. We show how samples lie on the embedding space using `CCSA - Second`. First, we

considered the row images of the MNIST and USPS datasets and plotted 2D visualization of them using t-SNE [171]. As Figure 2.5(Left) shows the row images of the same class and different domains lie far away from each other in the 2D subspace. For example, the samples of the class zero of the USPS dataset (0 U) are far from the class zero of the MNIST dataset (0 M). Second, we trained our base model with no adaptation on the MNIST dataset. We then plotted the 2D visualization of the MNIST and USPS samples in the embedding space (output of g, the last fully connected layer). As Figure 2.5(Middle) shows, the samples from the same class and different domains still lie far away from each other in the 2D subspace. Finally, we trained our SDA model on the MNIST dataset and 3 labeled samples per class of the USPS dataset. We then plotted the 2D visualization of the MNIST and USPS samples in the embedding space (output of g). As Figure 2.5(Right) shows, the samples from the same class and different domains now lie very close to each other in the 2D subspace. Note however, that this is only a 2D visualization of high-dimensional data, and Figure 2.5(Right) may not perfectly reflect how close is the data from the same class, and how classes are separated.

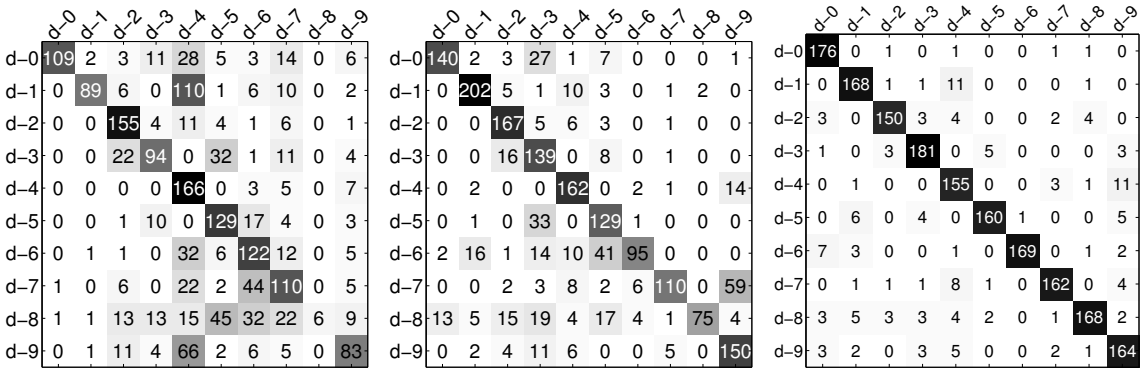

Figure 2.6: Confusion Matrices $\mathcal{U} \rightarrow \mathcal{M}$. Left: Confusion matrix for the baseline model when we train with source samples and test on target. The accuracy in this case is 59.05%. There are a lot of misclassified samples specifically for digit 8 (d-8). Middle: Confusion matrix for our model after adaptation when we used $n = 1$ target sample per class. The accuracy is 76.05% and with respect to the baseline model, there are less misclassified samples. Right: Confusion matrix for our model after adaptation when we used $n = 4$ target samples per class. The accuracy is 91.83% and there are very few misclassified samples.

Weight sharing: There is no restriction on whether or not g_t and g_s should share weights. Not sharing weights likely leads to overfitting, given the reduced amount of target training data, and weight-sharing acts as a regularizer. For instance, we repeated the experiment for the $\mathcal{M} \rightarrow \mathcal{U}$ task with $n = 4$ and the second choice. Not sharing weights provides an average accuracy of 89.1

over 10 repetitions, which is less than the average accuracy with weight-sharing (see Table 2.1). A similar behavior is observable in other experiments.

Confusion Matrices. We designed one simple experiment to see how the confusion matrices changes per class during adaptation. Similar to our previous experiments, we randomly selected 1800 samples from USPS dataset as source samples and 2000 samples from MNIST as target samples. We also consider $n = 1$ and $n = 4$ target samples per class in training during adaptation. Figure 2.6 shows that using adaptation provides very good results.

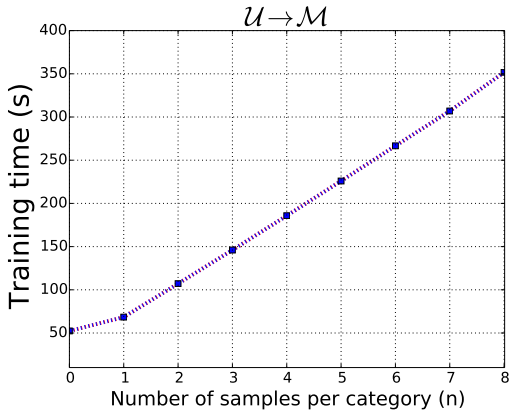

Figure 2.7: Training Time. Training time with respect to the number of samples per category. The number of created pairs linearly depends on the number of training target samples per category.

Training Time. Since we are creating pairs in training, The training time linearly increases if we use all training pairs. However, since we are not using all the negative pairs, training time will be slower than linear growth (Figure 2.7).

Combining Source and Target datasets. Finetuning is an effective method to transfer the knowledge from the source domain to the target domain. Another baseline could be simply aggregating all the samples from source and target datasets to one single dataset to train the model. After applying this method to several experiments, we observed that it provided poorer results than finetuning in the most cases.

For example, we repeated the experiment for the $\mathcal{M} \rightarrow \mathcal{U}$ task. The classification accuracy for $n = 4$, $n = 5$, and $n = 6$ is 84.6, 86.1, and 87.2, respectively, which are poorer than finetuning (86.5, 87.2, and 88.4).

Office Dataset

The office dataset is a standard benchmark dataset for visual domain adaptation. It contains 31 object classes for three domains: Amazon, Webcam, and DSLR, indicated as $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$, for a total of 4,652 images. We consider six domain shifts using the three domains ($\mathcal{A} \to \mathcal{W}$, $\mathcal{A} \to \mathcal{D}$, $\mathcal{W} \to \mathcal{A}$, $\mathcal{W} \to \mathcal{D}$, $\mathcal{D} \to \mathcal{A}$, and $\mathcal{D} \to \mathcal{W}$). We performed several experiments using this dataset.

First experiment. We followed the setting described in [62]. All classes of the office dataset and 5 train-test splits are considered. For the source domain, 20 examples per category for the Amazon domain, and 8 examples per category for the DSLR and Webcam domains are randomly selected for training for each split. Also, 3 labeled examples are randomly selected for each category in the target domain for training for each split. The rest of the target samples are used for testing. Note that we used the same splits generated by [62]. We also report the classification results of the SDA algorithm presented in [65], [63], and [78]. In addition to the SDA algorithms, we report the results of some recent UDA algorithms. They follow a different experimental protocol compared to the SDA algorithms, and use all samples of the target domain in training as unlabeled data together with all samples of the source domain.

For the embedding function g, we used the convolutional layers of the VGG-16 architecture [46] followed by 2 fully connected layers with output size of 1024 and 128, respectively. For the prediction function h, we used a fully connected layer with softmax activation. Similar to [62], we used the weights pre-trained on the ImageNet dataset [48] for the convolutional layers, and initialized the fully connected layers using all the source domain data. We then fine-tuned all the weights using the train-test splits.

CCSA - first, CCSA - second, and CCSA - third correspond to our model with discriminator, Contrastive, and Triplet choices, respectively.

Table 2.2 reports the classification accuracy over 31 classes for the Office dataset and shows that CCSA - Second (Contrastive choice) and CCSA - Third (Triplet choice) have better performance compared to [62]. Since the difference between $\mathcal{W}$ domain and $\mathcal{D}$ domain is not considerable, unsupervised algorithms work well on $\mathcal{D} \to \mathcal{W}$ and $\mathcal{W} \to \mathcal{D}$. However, in the cases when target and source domains are very different ($\mathcal{A} \to \mathcal{W}$, $\mathcal{W} \to \mathcal{A}$, $\mathcal{A} \to \mathcal{D}$, and $\mathcal{D} \to \mathcal{A}$), CCSA shows larger margins compared to the second best. This suggests that CCSA will provide greater alignment

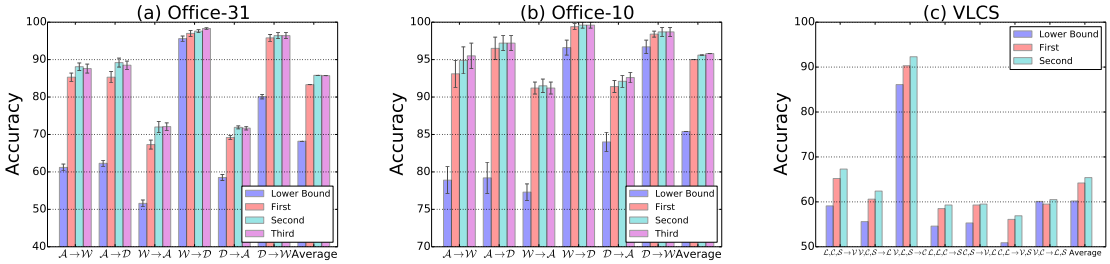

Figure 2.8: Improvement of CCSA over the base model.

gains when there are bigger domain shifts. Figure 2.8(a) instead, shows how much improvement can be obtained with respect to the base model. This is simply obtained by training g and h with only the classification loss and source training data, so no adaptation is performed.

Second experiment. We followed the setting described in [62] when only 10 target labeled samples of 15 classes of the Office dataset are available during training. Similar to [62], we compute the accuracy on the remaining 16 categories for which no target data was available during training. We used the same network structure as in the first experiment and the same splits generated by [62].

Table 2.3 shows that CCSA is effective at transferring information from the labeled classes to the unlabeled target classes. Similar to the first experiment, CCSA works well when shifts between domains are larger.

Third experiment. We used the original train-test splits of the Office dataset [13]. The splits are generated in a similar manner to the first experiment but here instead, only 10 classes are considered (backpack, bike, calculator, headphones, keyboard, laptop-computer, monitor, mouse, mug, and projector). In order to compare our results with the state-of-the-art, we used DeCaF-fc6 features [45] and 800-dimension SURF features as input. For DeCaF-fc6 features (SURF features) we used 2 fully connected layers with output size of 1024 (512) and 128 (32) with ReLU activation as the embedding function, and one fully connected layer with softmax activation as the prediction function. The features and splits are available on the Office dataset webpage [1].

We compared our results with three UDA (GFK [66], mSDA [166], and RTML [168]) and one SDA (CDML [167]) algorithms under the same settings. Table 2.4 shows that CCSA for all three choices provides an improved accuracy with respect to the others. Again, greater domain shifts

[1] `https://cs.stanford.edu/~jhoffman/domainadapt/`

are better compensated by CCSA . Figure 2.8(b) shows the improvement of CCSA over the base model using DeCaF-fc6 features. In this experiment, the third choice shows the better performance compare to the other two.

2.5.2 Domain Generalization

Since model CCSA - Third shows the similar performance compared to CCSA - second for domain generalization, we only evaluate CCSA - First and Second on different datasets. The goal is to show that CCSA is able to learn a domain invariant embedding subspace for visual recognition tasks.

2.5.3 VLCS Dataset

In this section, we use images of 5 shared object categories (bird, car, chair, dog, and person), of the PASCAL VOC2007 ($\mathcal{V}$) [172], LabelMe ($\mathcal{L}$) [61], Caltech-101 ($\mathcal{C}$) [173], and SUN09 ($\mathcal{S}$) [174] datasets, which is known as VLCS dataset [129].

[127, 126] have shown that there are covariate shifts between the above 4 domains and have developed a DG method to minimize them. We followed their experimental setting, and randomly divided each domain into a training set (70%) and a test set (30%) and conducted a `leave-one-domain-out` evaluation (4 cross-domain cases) and a `leave-two-domain-out` evaluation (3 cross-domain cases). In order to compare our results with the state-of-the-art, we used DeCaF-fc6 features which are publicly available [2], and repeated each cross-domain case 20 times and reported the average classification accuracy.

We used 2 fully connected layers with output size of 1024 and 128 with ReLU activation as the embedding function g, and one fully connected layer with softmax activation as the prediction function h. To create positive and negative pairs for training our network, for each sample of a source domain we randomly selected 5 samples from each remaining source domain, and help in this way to avoid overfitting. However, to train a deeper network together with convolutional layers, it is enough to create a large amount of positive and negative pairs.

We report comparative results in Table 2.5, where all DG methods work better than the base

[2] `http://www.cs.dartmouth.edu/~chenfang/proj_page/FXR_iccv13/index.php`

Table 2.6: MNIST dataset. Classification accuracy for domain generalization over the MNIST dataset and its rotated domains.

	CAE [175]	*MTAE [127]*	*CCSA - Second*
$M_{15^\circ}, M_{30^\circ}, M_{45^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M$	72.1	82.5	**85.1**
$M, M_{30^\circ}, M_{45^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M_{15^\circ}$	95.3	**96.3**	95.9
$M, M_{15^\circ}, M_{45^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M_{30^\circ}$	92.6	93.4	**94.6**
$M, M_{15^\circ}, M_{30^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M_{45^\circ}$	81.5	78.6	**83.2**
$M, M_{15^\circ}, M_{30^\circ}, M_{45^\circ}, M_{75^\circ} \rightarrow M_{60^\circ}$	92.7	94.2	**94.8**
$M, M_{15^\circ}, M_{30^\circ}, M_{45^\circ}, M_{60^\circ} \rightarrow M_{75^\circ}$	79.3	80.5	**82.5**
Average	85.5	87.5	**89.3**

model, emphasizing the need for domain generalization. Our DG methods have higher average performance. Also, note that in order to compare with the state-of-the-art DG methods, we only used 2 fully connected layers for our network and precomputed features as input. However, when using convolutional layers on row images, we expect our DG models to provide better performance. Figure 2.8(c) shows the improvement of our DG models over the base model using DeCaF-fc6 features.

2.5.4 MNIST Dataset

We followed the setting in [127], and randomly selected a set M of 100 images per category from the MNIST dataset (1000 in total). We then rotated each image in M five times with 15 degrees intervals, creating five new domains M_{15°, M_{30°, M_{45°, M_{60°, and M_{75°. We conducted a `leave-one-domain-out` evaluation (6 cross-domain cases in total). We used the same network of Section 2.5.1, and we repeated the experiments 10 times. To create positive and negative pairs for training our network, for each sample of a source domain we randomly selected 2 samples from each remaining source domain. We report comparative average accuracies for CCSA - Second and others in Table 2.6, showing again a performance improvement.

2.6 Discussion

The proposed method has a close relationship with Linear Discriminative Analysis (LDA) [176]. LDA looks for a projection where examples from the same class are projected very close to each other and, at the same time, the projected means are as far apart as possible. However, here we do not maximize between-class scatters of the samples in the same domain (source or target)

and we also do not minimize the within-class scatters of the samples in the same domain (source or target). We maximize the between-class scatters and within-class scatters of the cross domain samples. For the future work, it is interesting to use LDA for few-shot domain adaptation. It can be done by simply aggregating all available samples (from source and target) to one single dataset and applying LDA to that dataset.

2.7 Conclusions

We have introduced a deep model in combination with the classification and contrastive semantic alignment (CCSA) loss to address supervised domain adaptation (SDA) in the few-shot fashion. We have shown that the CCSA loss can be easily augmented to address the domain generalization (DG) problem without the need to change the basic model architecture. However, the approach is general in the sense that the architecture sub-components can be changed. We found that addressing the semantic distribution alignments with point-wise surrogates of distribution distances and similarities for SDA and DG works very effectively, even when labeled target samples are very few. We discussed three choices for point-wise surrogates of distribution distances and similarities and we showed that Triplet choice provides the best results among the three choices. In addition, we found the SDA accuracy to converge very quickly as more labeled target samples per category are available. The approach shows clear promise as it sets new state-of-the-art performance in all the experiments

Chapter 3

Few-Shot Adversarial Domain Adaptation

3.1 Introduction

As deep learning approaches have gained prominence in computer vision we have seen tasks that have large amounts of available labeled data flourish with improved results. There are still many problems worth solving where labeled data on an equally large scale is too expensive to collect, annotate, or both, and by extension a straightforward deep learning approach would not be feasible. Typically, in such a scenario, practitioners will train or reuse a model from a closely related dataset with a large amount of samples, here called the source domain, and then train with the much smaller dataset of interest, referred to as the target domain. This process is well-known under the name finetuning. Finetuning, while simple to implement, has been found to be sub-optimal when compared to later techniques such as *domain adaptation* [53]. Domain Adaptation can be *supervised* [62, 63], *unsupervised* [64, 65], or *semi-supervised* [66, 67, 68], depending on what data is available in a labeled format and how much can be collected.

Unsupervised domain adaptation (UDA) algorithms do not need any target data labels, but they require large amounts of target training samples, which may not always be available. Conversely, supervised domain adaptation (SDA) algorithms do require labeled target data, and because labeling information is available, for the same quantity of target data, SDA outperforms UDA [5]. Therefore, if the available target data is scarce, SDA becomes attractive, even if the labeling process is expensive, because only few samples need to be processed.

Most domain adaptation approaches try to find a feature space such that the confusion between source and target distributions in that space is maximum (*domain confusion*). Because of that, it is hard to say whether a sample in the feature space has come from the source distribution or the target distribution. Recently, generative adversarial networks [87] have been introduced for image generation which can also be used for domain adaptation. In [87], the goal is to learn a discriminator to distinguish between real samples and generated (fake) samples and then to learn a generator which best confuses the discriminator. Domain adaptation can also be seen as a generative adversarial network with one difference, in domain adaptation there is no need to generate samples, instead, the generator network is replaced with an inference network. Since the discriminator cannot determine if a sample is from the source or the target distribution the inference becomes optimal in terms of creating a joint latent space. In this manner, generative adversarial learning has been successfully modified for UDA [88, 89, 90] and provided very promising results.

Here instead, we are interested in adapting adversarial learning for SDA which we are calling *few-shot adversarial domain adaptation (FADA)* for cases when there are very few labeled target samples available in training. In this *few-shot learning* regime, our SDA method has proven capable of increasing a model's performance at a very high rate with respect to the inclusion of additional samples. Indeed, even one additional sample can significantly increase performance.

Our first contribution is to handle this scarce data while providing effective training. Our second contribution is to extend adversarial learning [87] to exploit the label information of target samples. We propose a novel way of creating pairs of samples using source and target samples to address the first challenge. We assign a group label to a pair according to the following procedure: 0 if samples of a pair come from the source distribution and the same class label, 1 if they come from the source and target distributions but the same class label, 2 if they come from the source distribution but different class labels, and 3 if they come from the source and target distributions and have different class labels. The second challenge is addressed by using adversarial learning [87] to train a deep inference function, which confuses a well-trained domain-class discriminator (DCD) while maintaining a high classification accuracy for the source samples. The DCD is a multi-class classifier that takes pairs of samples as input and classifies them into the above four groups. Confusing the DCD will encourage *domain confusion*, as well as the *semantic alignment* of classes. Our third contribution is an extensive validation of CCSA against the state-of-the-art. Although

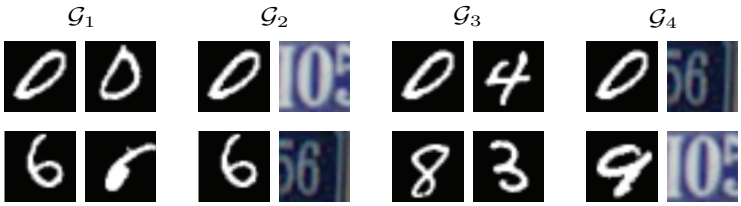

Figure 3.1: Examples from MNIST [6] and SVHN [7] of grouped sample pairs. $\mathcal{G}_1$ is composed of samples of the same class from the source dataset in this case MNIST. $\mathcal{G}_2$ is composed of samples of the same class, but one is from the source dataset and the other is from the target dataset. In $\mathcal{G}_3$ the samples in each pair are from the source dataset but with differing class labels. Finally, pairs in $\mathcal{G}_4$ are composed of samples from the target and source datasets with differing class labels.

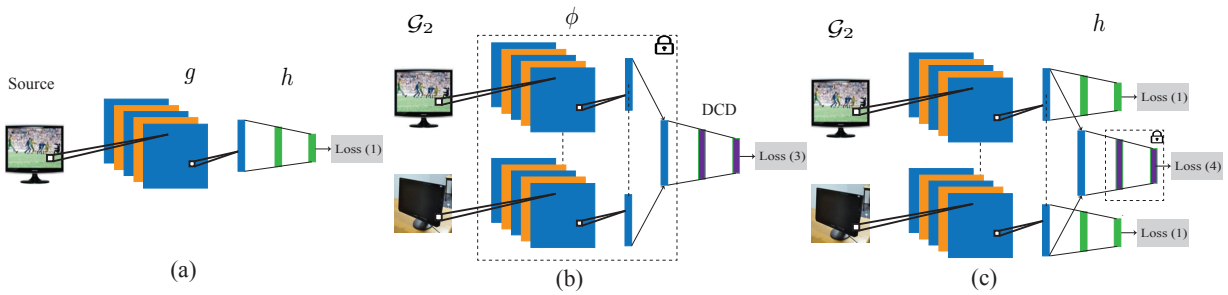

Figure 3.2: Few-shot adversarial domain adaptation. For simplicity we show our networks in the case of weight sharing ($g_s = g_t = g$). (a) In the first step, we initialized g and h using the source samples $\mathcal{D}_s$. (b) We freeze g and train a DCD. The picture shows a pair from the second group $\mathcal{G}_2$ when the samples come from two different distributions but the same class label. (c) We freeze the DCD and update g and h.

our method is general, and can be used for all domain adaptation applications, we focus on visual recognition.

3.2 Few-shot adversarial domain adaptation

In this section we describe the model we propose to address supervised domain adaptation (SDA). We are given a training dataset made of pairs $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^N$. The feature $x_i^s \in \mathcal{X}$ is a realization from a random variable X^s, and the label $y_i^s \in \mathcal{Y}$ is a realization from a random variable Y^s. In addition, we are also given the training data $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^M$, where $x_i^t \in \mathcal{X}$ is a realization from a random variable X^t, and the labels $y_i^t \in \mathcal{Y}$. We assume that there is a *covariate shift* [52] between X^s and X^t, i.e., there is a difference between the probability distributions $p(X^s)$ and $p(X^t)$. We say that X^s represents the *source domain* and that X^t represents the *target domain*. Under this settings the goal is to learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$ that during testing is going to perform well on data from the target domain.

Algorithm 1 FADA algorithm

1: Train g and h on $\mathcal{D}_s$ using (3.1).
2: Uniformly sample $\mathcal{G}_1$,$\mathcal{G}_3$ from $\mathcal{D}_s$x$\mathcal{D}_s$.
3: Uniformly sample $\mathcal{G}_2$,$\mathcal{G}_4$ from $\mathcal{D}_s$x$\mathcal{D}_t$.
4: Train DCD w.r.t. $g_t = g_s = g$ using (3.3).
5: **while** not convergent **do**
6: Update g and h by minimizing (3.5).
7: Update DCD by minimizing (3.3).
8: **end while**

The problem formulated thus far is typically referred to as *supervised domain adaptation*. In this work we are especially concerned with the version of this problem where only very few target labeled samples per class are available. We aim at handling cases where there is only one target labeled sample, and there can even be some classes with no target samples at all.

In absence of covariate shift a visual classifier f is trained by minimizing a *classification loss*

$$\mathcal{L}_C(f) = E[\ell(f(X^s), Y)] \,, \tag{3.1}$$

where $E[\cdot]$ denotes statistical expectation and ℓ could be any appropriate loss function. When the distributions of X^s and X^t are different, a deep model f_s trained with $\mathcal{D}_s$ will have reduced performance on the target domain. Increasing it would be trivial by simply training a new model f_t with data $\mathcal{D}_t$. However, $\mathcal{D}_t$ is small and deep models require large amounts of labeled data.

In general, f could be modeled by the composition of two functions, i.e., $f = h \circ g$. Here $g : \mathcal{X} \to \mathcal{Z}$ would be an inference from the input space $\mathcal{X}$ to a feature or inference space $\mathcal{Z}$, and $h : \mathcal{Z} \to \mathcal{Y}$ would be a function for predicting from the feature space. With this notation we would have $f_s = h_s \circ g_s$ and $f_t = h_t \circ g_t$, and the SDA problem would be about finding the best approximation for g_t and h_t, given the constraints on the available data.

If g_s and g_t are able to embed source and target samples, respectively, to a domain invariant space, it is safe to assume from the feature to the label space that $h_t = h_s = h$. Therefore, domain adaptation paradigms are looking for such inference functions so that they can use the prediction function h_s for target samples.

Traditional unsupervised DA (UDA) paradigms try to align the distributions of the features in the feature space, mapped from the source and the target domains using a metric between distributions, Maximum Mean Discrepancy [3] being a popular one and other metrics like Kullback Leibler divergence [177] and JensenShannon [87] divergence becoming popular when using adver-

sarial learning. Once they are aligned, a classifier function would no longer be able to tell whether a sample is coming from the source or the target domain. Recent UDA paradigms try to find inference functions to satisfy this important goal using adversarial learning. Adversarial training looks for a domain discriminator D that is able to distinguish between samples of source and target distributions. In this case D is a binary classifier trained with the standard cross-entropy loss

$$\mathcal{L}_{adv-D}(X_s, X_t, g_s, g_t) = -E[\log(D(g_s(X^s)))] - E[\log(1 - D(g_t(X^t)))] \; . \tag{3.2}$$

Once the discriminator is learned, adversarial learning tries to update the target inference function g_t in order to confuse the discriminator. In other words, the adversarial training is looking for an inference function g_t that is able to map a target sample to a feature space such that the discriminator D will no longer distinguish it from a source sample.

From the above discussion it is clear that in order to perform well, UDA needs to align the distributions effectively in order to be successful. This can happen only if distributions are represented by a sufficiently large dataset. Therefore, UDA approaches are in a position of weakness when we assume $\mathcal{D}_t$ to be small. Moreover, UDA approaches have also another intrinsic limitation; even with perfect confusion alignment, there is no guarantee that samples from different domains but with the same class label will map nearby in the feature space. This lack of *semantic alignment* is a major source of performance reduction.

3.2.1 Handling Scarce Target Data

We are interested in the case where very few labeled target samples (as low as 1 sample per class) are available. We are facing two challenges in this setting. First, since the size of $\mathcal{D}_t$ is small, we need to find a way to augment it. Second, we need to somehow use the label information of $\mathcal{D}_t$. Therefore, we create pairs of samples. In this way, we are able to alleviate the lack of training target samples by pairing them with each training source sample. In [5], we have shown that creating positive and negative pairs using source and target data is very effective for SDA. Since the method proposed in [5] does not encode the domain information of the samples, it cannot be used in adversarial learning. Here we extend [5] by creating 4 groups of pairs ($\mathcal{G}_i, i = 1, 2, 3, 4$) as follows: we break down the positive pairs into two groups (Groups 1 and 2), where pairs of the

first group consist of samples from the source distribution with the same class labels, while pairs of the second group also have the same class label but come from different distributions (one from the source and one from the target distribution). This is important because we can encode both label and domain information of training samples. Similarly, we break down the negative pairs into two groups (Groups 3 and 4), where pairs of the third group consist of samples from the source distribution with different class labels, while pairs of the forth group come from different class labels and different distributions (one from the source and one from the target distributions). See Figure 3.1. In order to give each group the same amount of members we use all possible pairs from $\mathcal{G}_2$, as it is the smallest, and then uniformly sample from the pairs in $\mathcal{G}_1$, $\mathcal{G}_3$, and $\mathcal{G}_4$ to match the size of $\mathcal{G}_2$. Any reasonable amount of portions between the numbers of the pairs can also be used.

In classical adversarial learning we would at this point learn a domain discriminator, but since we have semantic information to consider as well, we are interested in learning a multi-class discriminator (we call it domain-class discriminator (DCD)) in order to introduce *semantic alignment* of the source and target domains. By expanding the binary classifier to its multiclass equivalent, we can train a classifier that will evaluate which of the 4 groups a given sample pair belongs to. We model the DCD with 2 fully connected layers with a softmax activation in the last layer which we can train with the standard categorical cross-entropy loss

$$\mathcal{L}_{FADA-D} = -E[\sum_{i=1}^{4} y_{\mathcal{G}_i} \log(D(\phi(\mathcal{G}_i)))] \,, \tag{3.3}$$

where $y_{\mathcal{G}_i}$ is the label of $\mathcal{G}_i$ and D is the DCD function. ϕ is a symbolic function that takes a pair as input and outputs the concatenation of the results of the appropriate inference functions. The output of ϕ is passed to the DCD (Figure 3.2).

In the second step, we are interested in updating g_t in order to confuse the DCD in such a way that the DCD can no longer distinguish between groups 1 and 2, and also between groups 3 and 4 using the loss

$$\mathcal{L}_{FADA-g} = -E[y_{\mathcal{G}_1} \log(D(\phi(\mathcal{G}_2))) + y_{\mathcal{G}_3} \log(D(\phi(\mathcal{G}_4)))] \,. \tag{3.4}$$

(3.4) is inspired by the non-saturating game [178] and will force the inference function g_t to embed target samples in a space that DCD will no longer be able to distinguish between them.

Connection with multi-class discriminators: Consider an image generation task where training samples come from k classes. Learning the image generator can be done by any standard k-class classifier and adding generated samples as a new class (generated class) and correspondingly increasing the dimension of the classifier output from k to $k+1$. During the adversarial learning, only the generated class is confused. This has proven effective for image generation [179] and other tasks. However, this is different than the proposed DCD, where group 1 is confused with 2, and group 3 is confused with 4. Inspired by [179], we are able to create a $k+4$ classifier to also guarantee a high classification accuracy. Therefore, we suggest that (3.4) needs to be minimized together with the main classifier loss

$$\mathcal{L}_{FADA-g} = -\gamma E[y_{\mathcal{G}_1} \log(D(g(\mathcal{G}_2))) + y_{\mathcal{G}_3} \log(D(g(\mathcal{G}_4)))] + E[\ell(f(X^s), Y)] + E[\ell(f(X^t), Y)] \ , \quad (3.5)$$

where γ strikes the balance between classification and confusion. Misclassifying pairs from group 2 as group 1 and likewise for groups 4 and 3, means that the DCD is no longer able to distinguish positive or negative pairs of different distributions from positive or negative pairs of the source distribution, while the classifier is still able to discriminate positive pairs from negative pairs. This simultaneously satisfies the two main goals of SDA, domain confusion and class separability in the feature space. UDA only looks for domain confusion and does not address class separability, because of the lack of labeled target samples.

Connection with conditional GANs: Concatenation of outputs of different inferences has been done before in conditional GANs. For example, [180, 181, 182] concatenate the input text to the penultimate layers of the discriminators. [183] concatenates positive and negative pairs before passing them to the discriminator. However, all of them use the vanilla binary discriminator.

Relationship between g_s and g_t: There is no restriction for g_s and g_t and they can be constrained or unconstrained. An obvious choice of constraint is equality (weight-sharing) which makes the inference functions symmetric. This can be seen as a regularizer and will reduce overfitting [5]. Another approach would be learning an asymmetric inference function [118]. Since we have access to very few target samples, we use weight-sharing ($g_s = g_t = g$).

Table 3.1: MNIST-USPS-SVHN datasets. Classification accuracy for domain adaptation over the MNIST, USPS, and SVHN datasets. $\mathcal{M}$, $\mathcal{U}$, and $\mathcal{S}$ stand for MNIST, USPS, and SVHN domain. LB is our base model without adaptation. FT and FADA stand for fine-tuning and our method, respectively.

		Traditional UDA			Adversarial UDA									
	LB	[165]	[118]	[64]	[88]	[89]	[90]	SDA	1	2	3	4	5	6	7
$\mathcal{M} \to \mathcal{U}$	65.4	47.8	60.7	91.8	91.2	89.4	92.5	FT	82.3	84.9	85.7	86.5	87.2	88.4	88.6
								[5]	85.0	89.0	90.1	91.4	92.4	93.0	92.9
								FADA	89.1	91.3	91.9	93.3	93.4	94.0	**94.4**
$\mathcal{U} \to \mathcal{M}$	58.6	63.1	67.3	73.7	89.1	90.1	90.8	FT	72.6	78.2	81.9	83.1	83.4	83.6	84.0
								[5]	78.4	82.2	85.8	86.1	88.8	89.6	89.4
								FADA	81.1	84.2	87.5	89.9	91.1	91.2	**91.5**
$\mathcal{S} \to \mathcal{M}$	60.1	-	-	82.0	76.0	-	84.7	FT	65.5	68.6	70.7	73.3	74.5	74.6	75.4
								FADA	72.8	81.8	82.6	85.1	86.1	86.8	**87.2**
$\mathcal{M} \to \mathcal{S}$	20.3	-	-	40.1	-	-	36.4	FT	29.7	31.2	36.1	36.7	38.1	38.3	39.1
								FADA	37.7	40.5	42.9	46.3	46.1	46.8	**47.0**
$\mathcal{S} \to \mathcal{U}$	66.0	-	-	-	-	-	-	FT	69.4	71.8	74.3	76.2	78.1	77.9	78.9
								FADA	78.3	83.2	85.2	85.7	86.2	87.1	**87.5**
$\mathcal{U} \to \mathcal{S}$	15.3	-	-	-	-	-	-	FT	19.9	22.2	22.8	24.6	25.4	25.4	25.6
								FADA	27.5	29.8	34.5	36.0	37.9	41.3	**42.9**

Choice of g_s, g_t, and h: Since we are interested in visual recognition, the inference functions g_s and g_t are modeled by a convolutional neural network (CNN) with some initial convolutional layers, followed by some fully connected layers which are described specifically in the experiments section. In addition, the prediction function h is modeled by fully connected layers with a softmax activation function for the last layer.

Training Process: Here we discuss the training process for the weight-sharing regularizer ($g_s = g_t = g$). Once the inference functions g and the prediction function h are chosen, FADA takes the following steps: First, g and h are initialized using the source dataset $\mathcal{D}_s$. Then, the mentioned four groups of pairs should be created using $\mathcal{D}_s$ and $\mathcal{D}_t$. The next step is training DCD using the four groups of pairs. This should be done by freezing g. In the next step, the inference function g and prediction function h should be updated in order to confuse DCD and maintain high classification accuracy. This should be done by freezing DCD. See Algorithm 1 and Figure 3.2. The training process for the non weight-sharing case can be derived similarly.

3.3 Experiments

We present results using the Office dataset [13], the MNIST dataset [6], the USPS dataset [169], and the SVHN dataset [7].

3.3.1 MNIST-USPS-SVHN Datasets

The MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), and SVHN ($\mathcal{S}$) datasets have recently been used for domain adaptation [170, 118, 89]. They contain images of digits from 0 to 9 in various different environments including in the wild in the case of SVHN [7]. We considered six cross-domain tasks. The first two tasks include $\mathcal{M} \rightarrow \mathcal{U}$, $\mathcal{U} \rightarrow \mathcal{M}$, and followed the experimental setting in [170, 118, 88, 89, 90], which involves randomly selecting 2000 images from MNIST and 1800 images from USPS. For the rest of the cross-domain tasks, $\mathcal{M} \rightarrow \mathcal{S}$, $\mathcal{S} \rightarrow \mathcal{M}$, $\mathcal{U} \rightarrow \mathcal{S}$, and $\mathcal{S} \rightarrow \mathcal{U}$, we used all training samples of the source domain for training and all testing samples of the target domain for testing.

Since [170, 118, 88, 89, 90] introduced unsupervised methods, they used all samples of a target domain as unlabeled data in training. Here instead, we randomly selected n labeled samples per class from target domain data and used them in training. We evaluated our approach for n ranging from 1 to 4 and repeated each experiment 10 times (we only show the mean of the accuracies for this experiment because standard deviation is very small).

Since the images of the USPS dataset have 16×16 pixels, we resized the images of the MNIST and SVHN datasets to 16×16 pixels. We assume g_s and g_t share weights ($g = g_s = g_t$) for this experiment. Similar to [6], we used 2 convolutional layers with 6 and 16 filters of 5×5 kernels followed by max-pooling layers and 2 fully connected layers with size 120 and 84 as the inference function g, and one fully connected layer with softmax activation as the prediction function h. Also, we used 2 fully connected layers with size 64 and 4 as DCD (4 groups classifier). Training for each stage was done using the Adam Optimizer [184]. We compare our method with 1 SDA method, under the same condition, and 6 recent UDA methods. UDA methods use all target samples in their training stage, while we only use very few labeled target samples per category in training.

Table 3.1 shows the classification accuracies across a range for the number of target samples available in training ($n = 1, \ldots, 7$). CCSA works well even when only one target sample per category ($n = 1$) is available in training. We can get comparable accuracies with the state-of-the-art using only 10 labeled target samples (one sample per class $n = 1$) instead of using more than thousands of unlabeled target samples. We also report the lower bound (`LB`) of our model which corresponds to training the base model using only source samples. Moreover, we report the accuracies obtained by fine-tuning (`FT`) the base model on available target data and also the recent

Table 3.2: Office dataset. Classification accuracy for domain adaptation over the 31 categories of the Office dataset. $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$ stand for Amazon, Webcam, and DSLR domain. LB is our base model without adaptation.

		Unsupervised Methods			Supervised Methods			
	LB	[165]	[65]	[64]	[62]	[63]	[5]	CCSA
$\mathcal{A} \to \mathcal{W}$	61.2 ± 0.9	61.8 ± 0.4	68.5 ± 0.4	68.7 ± 0.3	82.7 ± 0.8	84.5 ± 1.7	**88.2 ± 1.0**	88.1 ± 1.2
$\mathcal{A} \to \mathcal{D}$	62.3 ± 0.8	64.4 ± 0.3	67.0 ± 0.4	67.1 ± 0.3	86.1 ± 1.2	86.3 ± 0.8	**89.0 ± 1.2**	88.2 ± 1.0
$\mathcal{W} \to \mathcal{A}$	51.6 ± 0.9	52.2 ± 0.4	53.1 ± 0.3	54.09 ± 0.5	65.0 ± 0.5	65.7 ± 1.7	**72.1 ± 1.0**	71.1 ± 0.9
$\mathcal{W} \to \mathcal{D}$	95.6 ± 0.7	98.5 ± 0.4	**99.0 ± 0.2**	**99.0 ± 0.2**	97.6 ± 0.2	97.5 ± 0.7	97.6 ± 0.4	97.5 ± 0.6
$\mathcal{D} \to \mathcal{A}$	58.5 ± 0.8	52.1 ± 0.8	54.0 ± 0.4	56.0 ± 0.5	66.2 ± 0.3	66.5 ± 1.0	**71.8 ± 0.5**	68.1 ± 06
$\mathcal{D} \to \mathcal{W}$	80.1 ± 0.6	95.0 ± 0.5	96.0 ± 0.3	**96.4 ± 0.3**	95.7 ± 0.5	95.5 ± 0.6	**96.4 ± 0.8**	**96.4 ± 0.8**
Average	68.2	70.6	72.9	73.6	82.2	82.6	**85.8**	84.9

work presented in [5]. Although Table 3.1 shows that FT increases the accuracies over LB, it has reduced performance compared to SDA methods.

Figure 3.3 shows how much improvement can be obtained with respect to the base model. The base model is the lower bound LB. This is simply obtained by training g and h with only the classification loss and source training data; so, no adaptation is performed.

Weight-Sharing. As we discussed earlier, weight-sharing can be seen as a regularizer that prevents the target network g_t from overfitting. This is important because g_t can be easily overfitted since target data is scarce. We repeated the experiment for the $\mathcal{U} \to \mathcal{M}$ with $n = 5$ without sharing weights. This provides an average accuracy of 84.1 over 10 repetitions, which is less than the weight-sharing case.

3.3.2 Office Dataset

The office dataset is a standard benchmark dataset for visual domain adaptation. It contains 31 object classes for three domains: Amazon, Webcam, and DSLR, indicated as $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$, for a total of 4,652 images. The first domain $\mathcal{A}$, consists of images downloaded from online merchants, the second $\mathcal{W}$, consists of low resolution images acquired by webcams, the third $\mathcal{D}$, consists of high resolution images collected with digital SLRs. We consider four domain shifts using the three domains ($\mathcal{A} \to \mathcal{W}$, $\mathcal{A} \to \mathcal{D}$, $\mathcal{W} \to \mathcal{A}$, and $\mathcal{D} \to \mathcal{A}$). Since there is not a considerable domain shift between $\mathcal{W}$ and $\mathcal{D}$, we exclude $\mathcal{W} \to \mathcal{D}$ and $\mathcal{D} \to \mathcal{W}$.

We followed the setting described in [62]. All classes of the office dataset and 5 train-test splits are considered. For the source domain, 20 examples per category for the Amazon domain, and 8 examples per category for the DSLR and Webcam domains are randomly selected for training for each split. Also, 3 labeled examples are randomly selected for each category in the target domain

for training for each split. The rest of the target samples are used for testing. Note that we used the same splits generated by [62].

In addition to the SDA algorithms, we report the results of some recent UDA algorithms. They follow a different experimental protocol compared to the SDA algorithms, and use all samples of the target domain in training as unlabeled data together with all samples of the source domain. So, we cannot make an exact comparison between results. However, since UDA algorithms use all samples of the target domain in training and we use only very few of them (3 per class), we think it is still worth looking at how they differ.

Here we are interested in the case where g_s and g_t share weights ($g_s = g_t = g$). For the inference function g, we used the convolutional layers of the VGG-16 architecture [46] followed by 2 fully connected layers with output size of 1024 and 128, respectively. For the prediction function h, we used a fully connected layer with softmax activation. Similar to [62], we used the weights pre-trained on the ImageNet dataset [48] for the convolutional layers, and initialized the fully connected layers using all the source domain data. We model the DCD with 2 fully connected layers with a softmax activation in the last layer.

Table 3.2 reports the classification accuracy over 31 classes for the Office dataset and shows that CCSA has performance comparable to the state-of-the-art.

3.4 Conclusions

We have introduced a deep model combining a classification and an adversarial loss to address SDA in few-shot learning regime. We have shown that adversarial learning can be augmented to address SDA. The approach is general in the sense that the architecture sub-components can be changed. We found that addressing the semantic distribution alignments with point-wise surrogates of distribution distances and similarities for SDA works very effectively, even when labeled target samples are very few. In addition, we found the SDA accuracy to converge very quickly as more labeled target samples per category are available. The approach shows clear promise as it sets new state-of-the-art performance in the experiments.

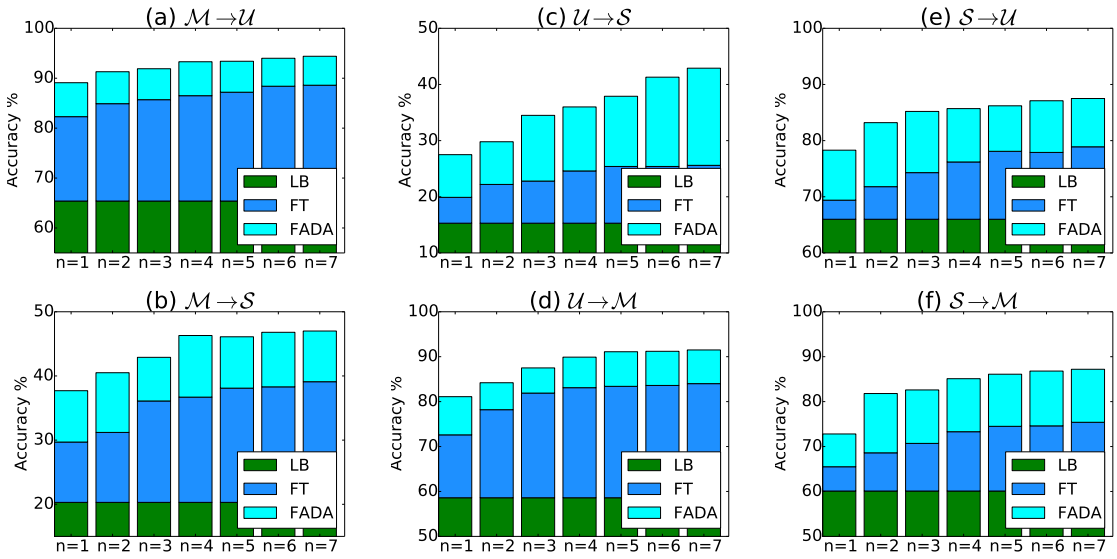

Figure 3.3: MNIST-USPS-SVHN summary. The lower bar of each column represents the `LB` as reported in Table 3.1 for the corresponding domain pair. The middle bar is the improvement of fine-tuning `FT` the base model using the available target data reported in Table 3.1. The top bar is the improvement of `FADA` over `FT`, also reported in Table 3.1.

Chapter 4

Information Bottleneck Learning Using Privileged Information

4.1 Introduction

We address the auxiliary view problem from an information theoretic perspective, where we learn how to extract information from the main data view, in a way that is optimal for visual recognition, and that speaks also on behalf of the missing auxiliary view [79]. The information bottleneck (IB) method [8] is a tool for extracting *latent information* from the main view, in a way that satisfies two complementary goals. The first is to compress the data as much as possible. The second is to preserve all the information that is *relevant* for the task at hand (e.g., predicting the labels of a visual recognition task). However, the IB method is not directly applicable to our problem because the latent information is not extracted in a way that speaks also on behalf of the auxiliary view. Therefore, our first contribution is to extend the IB method to take that aspect into account. Since the auxiliary view is not available at testing time, it was named *privileged* in [56], which first formalized this learning paradigm. Thus, we refer to our IB extension as the *information bottleneck method with privileged information (IBPI)*.

The IBPI method is a sound information theoretic principle for explicitly extracting relevant latent information, but gives an implicit, hence computationally hard, way for learning a visual classifier based on such information. Our second contribution is a modified version of IBPI that allows learning explicitly any type of visual classifier based on risk minimization. Our third contribution is the application of the modified IBPI method for learning a large-margin classifier, called

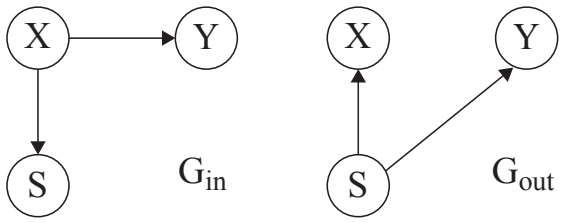

Figure 4.1: Information Bottleneck. Structural representation of G_{in} and G_{out} used by the original two-variable information bottleneck method [8].

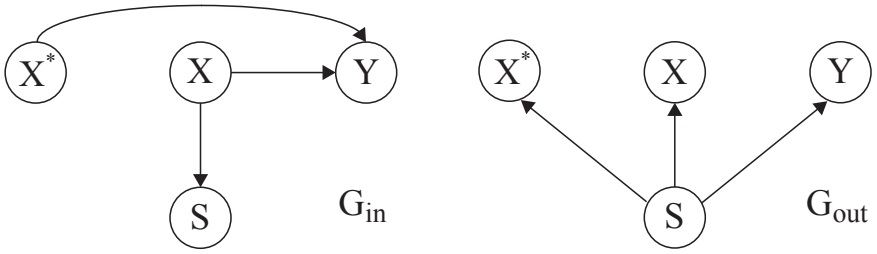

Figure 4.2: Information Bottleneck with Privileged Information. Structural representation of G_{in} and G_{out} used by the information bottleneck method with privileged information.

large-margin IBPI (LMIBPI), for which it is possible to use kernels, and for which we provide an optimization procedure guaranteed to converge in the primal space for improved computational efficiency.

Our fourth contribution is an extensive validation of LMIBPI against the state-of-the-art. We perform experiments where we improve visual recognition of gestures by training with auxiliary 3D joint information, we improve object classification with auxiliary object bounding box information, we improve animal recognition with auxiliary attribute information, and we improve action recognition with auxiliary visual features.

4.1.1 Problem statement

Traditional supervised learning assumes that a training dataset made of N pairs $(x_1, y_1), \cdots, (x_N, y_N)$ is given, where the feature $x_i \in \mathcal{X}$ is a realization from a random variable X, the label $y_i \in \mathcal{Y}$ is a realization from a random variable Y, and the pairs are i.i.d. samples from a joint probability distribution $p(X, Y)$. Under this setting the goal is to learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$ by searching over a space of admissible functions $\mathcal{F}$.

The Learning Using Privileged Information (LUPI) paradigm as defined in [56] assumes that every training data pair comes with *auxiliary* information, augmenting the training dataset to $(x_1, x_1^*, y_1), \cdots, (x_N, x_N^*, y_N)$. The auxiliary feature $x_i^* \in \mathcal{X}^*$ is a realization from the random

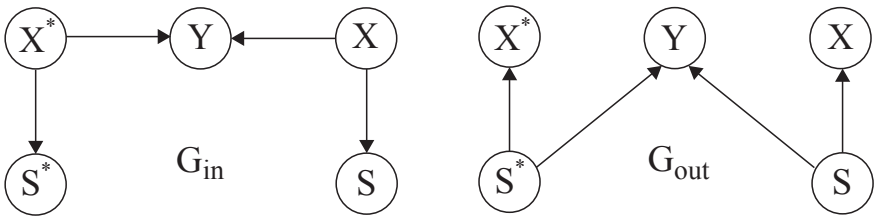

Figure 4.3: Upper-bound IBPI. Structural representation of G_{in} and G_{out} used by the information bottleneck method when main and auxiliary information are fused to provide the upper-bound of the IBPI method.

variable X^*. The triplets are now i.i.d. samples from the joint distribution $p(X, X^*, Y)$. Under LUPI settings, the goal is to learn a prediction function $f^* : \mathcal{X} \to \mathcal{Y}$ by searching $\mathcal{F}$. Note that in order to predict a label y, at testing time f^* uses only data from the *main* space $\mathcal{X}$. Therefore, the data from the auxiliary space $\mathcal{X}^*$ is only available during training, which is why it is called *privileged*. From the same amount of training samples N, the LUPI classifier f^* will improve the performance of the traditional classifier f [136]. On the other hand, how to *best* exploit privileged information for learning f^* remains an open problem.

4.2 The information bottleneck method

We summarize the information bottleneck (IB) method [8] that was extended to the multivariate case in [185]. We are given a set of random variables $\mathbf{X} = \{X_1, \cdots, X_n\}$, distributed according to a known $p(\mathbf{X})$, a set of *latent* variables $\mathbf{S} = \{S_1, \cdots, S_k\}$, and a Bayesian network with graph G_{in} over $\mathbf{X} \cup \mathbf{S}$, defining which subset of $\mathbf{X}$ is compressed by which subset of $\mathbf{S}$.

Another Bayesian network, G_{out}, also defined over $\mathbf{X} \cup \mathbf{S}$, is given and represents which conditional dependencies and independencies we desire $\mathbf{S}$ to be able to generate. The joint distribution $q(\mathbf{X}, \mathbf{S}) \doteq q(\mathbf{S}|\mathbf{X})p(\mathbf{X})$ is unknown.

The compression requirements defined by G_{in}, and the desired independencies defined by G_{out}, are incompatible in general. Therefore, *the multivariate IB method computes the optimal* $\mathbf{S}$ *by searching for the distribution* $q(\mathbf{S}|\mathbf{X})$*, where* $\mathbf{S}$ *compresses* $\mathbf{X}$ *as much as possible, while the distance from* $q(\mathbf{X}, \mathbf{S})$ *to the closest distribution among those consistent with the structure of* G_{out} *is minimal.* This idea is implemented with the *multi-information* of $\mathbf{X}$, which is the information shared by $X_1, \cdots, X_n$, i.e.,

$$\mathcal{I}(\mathbf{X}) = D_{KL}[p(\mathbf{X}) \| p(X_1) \cdots p(X_n)] \,, \tag{4.1}$$

where D_{KL} indicates the Kullback-Leibler divergence [186]. Therefore, the multivariate IB method looks for $q(\mathbf{S}|\mathbf{X})$ that minimizes the functional

$$\mathcal{L}[q(\mathbf{S}|\mathbf{X})] = \mathcal{I}^{G_{in}}(\mathbf{X},\mathbf{S}) + \gamma(\mathcal{I}^{G_{in}}(\mathbf{X},\mathbf{S}) - \mathcal{I}^{G_{out}}(\mathbf{X},\mathbf{S})) \tag{4.2}$$

where γ strikes a balance between compression and the ability to satisfy the independency requirements of G_{out}. The multi-information $\mathcal{I}^G$ with respect to a Bayesian network G defined over $\mathbf{X} \sim p(\mathbf{X})$ is computed as in [185], i.e.,

$$\mathcal{I}^G(\mathbf{X}) = \sum_i I(X_i; \mathbf{Pa}^G_{X_i}) \ , \tag{4.3}$$

where $I(X_i; \mathbf{Pa}^G_{X_i})$ is the mutual information between X_i and $\mathbf{Pa}^G_{X_i}$, the set of variables that are parents of X_i in G.

Let us refer to Figure 4.1 for an example, where $\mathbf{X} = \{X, Y\}$, and $\mathbf{S} = S$. We interpret X as the *main data* we want to compress, and from which we would like to predict the *relevant information* Y. This is achieved by first compressing X into S, and then predicting Y from S. In G_{in} the edge $X \to Y$ indicates the relation defined by $p(X, Y)$. Moreover, since S will compress X, this is indicated by the edge $X \to S$, establishing that S is completely determined given the variable it compresses. The graph G_{out} instead, reflects the idea that we would like S to capture from X all the necessary information to perform the best possible prediction of Y. This means that knowing S makes X and Y independent, or equivalently that $I(X; Y|S) = 0$.

To evaluate (4.2), instead, we obtain $\mathcal{I}^{G_{in}} = I(S; X) + I(Y; X)$, and $\mathcal{I}^{G_{out}} = I(X; S) + I(Y; S)$, and since $I(Y; X)$ is constant, (4.2) collapses to the original two-variable IB method [8].

4.3 IB with privileged information

Here we combine the ideas of Sections 4.1 and 4.2 for developing a new information bottleneck principle, which accounts for privileged information. Specifically, let us assume that X, X^*, and Y are three random variables with known distribution $p(X, X^*, Y)$. Also, it is assumed that both X and X^* contain information about Y. If properly extracted, such information could be used for predicting Y. However, we assume that only the information carried by X can be used to predict Y. We pose the question of whether by doing so it is still possible to learn a model capable of

exploiting the information carried by X^*.

If we apply the two-variable IB method, we proceed by compressing X into a latent variable S as much as possible, while making sure that information about Y is retained. These two competing goals are depicted by the two graphs G_{in} and G_{out} in Figure 4.1. On the other hand, since X^* has knowledge about Y, a more complete Bayesian network representing all the variables and the compression requirements, is the graph G_{in} in Figure 4.2, which includes the connection $X^* \to Y$. Therefore, the optimal representation computed by the two-variable IB method would be given by $q(X, X^*, Y, S) = q(S|X)p(X, X^*, Y)$, where $q(S|X)$ is such that $I(X;Y|S)$ is as close to zero as possible.

We note that the approach outlined above does not make any effort to exploit the information carried by X^*. Indeed, $I(X^*;Y|S)$ could be arbitrarily high, i.e., knowing S still leaves with X^* substantial knowledge about Y. On the other hand, the multivariate IB method allows us to consider more complex independency structures. In particular, we define G_{out} like in Figure 4.2, where knowing S not only makes X and Y independent, but X^* and Y too. In this way, $q(S|X)$ not only minimizes $I(X;Y|S)$, but also $I(X^*;Y|S)$. More precisely, the multi-informations of G_{in} and G_{out} in Figure 4.2 are given by

$$\mathcal{I}^{G_{in}} = I(S;X) + I(Y;X,X^*)\ , \tag{4.4}$$

$$\mathcal{I}^{G_{out}} = I(S;X) + I(S;X^*) + I(S;Y)\ . \tag{4.5}$$

By plugging (4.4) and (4.5) into (4.2), since $I(Y;X,X^*)$ is constant, the functional for learning the optimal representation for S is given by

$$\boxed{\mathcal{L}[q(S|X)] = I(S;X) - \gamma I(X^*;S) - \gamma I(S;Y)} \tag{4.6}$$

where γ strikes a balance between compressing X and imposing the independency requirements. Similarly to the LUPI framework, since it is not possible to directly compress X^* for predicting Y, we can think of X^* as carrying *privileged information* about Y. Therefore, we call learning representations by minimizing (4.6) as the *information bottleneck method with privileged information (IBPI)*.

4.4 IBPI for visual recognition

We are interested in designing a framework for visual recognition, where we need to perform a classification task based on a *main* view X of the visual data. However, at training time, for some training samples an *auxiliary* view X^* is also available. We pose no restrictions on the type of auxiliary data available. The task at hand falls into the LUPI category defined in Section 4.1, except that we also admit training samples with missing auxiliary view.

We want to leverage the IBPI method (4.6) because it provides a sound principle, grounded on information theory, for extracting information S from the main view X that is not only the most relevant for predicting Y (representing class labels), but also minimizes $I(X^*;Y|S)$, which means that knowing S leaves with X^* minimal information about Y. This suggests that S is the representation of choice for predicting Y. However, similarly to the IB method [185], while IBPI explicitly defines the compression map, S, by searching for $q(S|X)$, the computation of $q(Y|S)$ is much harder in general. For this reason, we introduce a modified IBPI method that is tailored to visual recognition.

We observe that by interpreting γ as a Lagrange multiplier, the last term in (4.6) corresponds to the constraint $I(S;Y) \geq \textit{constant}$, enforcing S of carrying at least a certain amount of information about Y. Ultimately, such information should be used for classification purposes, by predicting Y through a function $\tilde{f} : \mathcal{S} \rightarrow \mathcal{Y}$. Therefore, we replace the constraint on $I(S;Y)$ with the risk associated to $\tilde{f}(S)$ according to a loss function ℓ. Thus, for visual recognition, (4.6) is modified into

$$\boxed{\mathcal{L}[q(S|X), \tilde{f}] = I(S;X) - \gamma I(X^*;S) + \beta E[\ell(\tilde{f}(S), Y)]} \tag{4.7}$$

where $E[\cdot]$ denotes statistical expectation, and β balances the risk versus the compression requirements. Note that the modified IBPI criterion (4.7) is general, and could be used with any classifier. Obviously, a practical implementation of (4.7) would be based on the empirical risk.

4.4.1 Large-margin IBPI

We use (4.7) to develop a large-margin classifier. We focus on the binary case to prove the validity of the framework by comparing it with the state-of-the-art, which also focused on the binary case. In particular, we restrict the search space for $q(S|X)$ by assuming $S = \phi(X;A)$,

Algorithm 2 Projected gradient minimization for F

Chose $0 < \eta < 1$, $0 < \nu < 1$.
Initialize F^1. Set $\rho = 1$.
for $k = 1, 2, \cdots$ **do**
  if ρ satisfies (4.11) **then**
    Repeatedly increase it by $\rho \leftarrow \rho/\eta$ until either ρ does not satisfy (4.11) or $F(\rho/\eta) = F(\rho)$
  else
    Repeatedly decrease ρ by $\rho \leftarrow \rho/\eta$ until ρ satisfies (4.11)
  end if
  Set $F^{k+1} = \max\{0, F^k - \rho \nabla_F D_{KL}(\bar{X} \| F^k \bar{X}^*)\}$
  Normalize to 1 the columns of F^{k+1}
end for

Algorithm 3 FALM for LMIBPI

Chose $\mu_f > 0$ and $\mu_g > 0$ and $A^0 = B^0 = E^1$, set $t_1 = 1$
for $k = 1, 2, \cdots$ **do**
  $A^k = \arg\min_{0 \le A \le 1} Q_g(A, E^k)$
  $B^k = \arg\min_{0 \le B \le 1} Q_f(B, A^k)$
  $t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2$
  $E^{k+1} = B^k + \frac{t_k - 1}{t_{k+1}}(B^k - B^{k-1})$
end for

where A is a suitable set of parameters. Moreover, $\tilde{f}(S)$ is a binary decision function given by $Y = \text{sign}(\langle w, S \rangle + b)$, where $\langle \cdot, \cdot \rangle$ identifies a dot product, w defines the margin, and b is an offset. Therefore, by using the hinge loss function, from (4.7) we derive the following classifier learning formulation, which we refer to as the *large-margin IBPI (LMIBPI)*

$$\begin{aligned} \min_{A,w,b,\xi_i} \; & I(S;X) - \gamma I(X^*;S) + \frac{\beta}{2}\|w\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(\langle w, \phi(x_i, A)\rangle + b) \ge 1 - \xi_i \;, \\ & \xi_i \ge 0 \;, \quad \forall i \in \{1, \cdots, N\} \;. \end{aligned} \tag{4.8}$$

where C is the usual parameter to control the slackness.

Kernels. We set $S = \phi(X, A) = A\phi(X)$, where we require $\phi(X)$ to have positive components and be normalized to 1, and A to be a stochastic matrix, made of conditional probabilities between components of $\phi(X)$ and S. This assumption greatly simplifies computing mutual informations. X can be mapped to a feature space with $\psi(X)$. In this case we set $\phi(X) = \rho(\Psi\psi(X))^\top$, where $\Psi = [\psi(x_1), \cdots, \psi(x_N)]$, and $\rho(\cdot)$ is the additive logistic transformation that maps $u \in \mathbb{R}^N$ to the $N+1$ dimensional simplex $v = \left[\frac{e^{u_1}}{1+\sum_i e^{u_i}}, \cdots, \frac{e^{u_N}}{1+\sum_i e^{u_i}}, \frac{1}{1+\sum_i e^{u_i}}\right]$, with positive components and normalized to 1. Thus, without loss of generality, in the sequel we set $S = AX$. X^* can be mapped to a feature space $\varphi(X^*)$ with the same strategy.

Mutual informations. $I(S;X)$ is given by

$$I(S;X) = E\left[\sum_{i,j} A(i,j)X(j)\log\frac{A(i,j)}{S(i)}\right] \tag{4.9}$$

where $A(i,j)$ is the entry of A in position i,j, whereas $S(i)$ and $X(j)$ are the components in position i and j of S and X respectively. Obviously, during training the expectation is replaced by the empirical average.

To compute $I(S;X^*)$, let $s(i)$, $x^*(j)$, and $x(h)$ be histogram realizations for S, X^*, and X, where i, j, and h index the histogram bins. The mutual information $I(t,x^*)$ is given by $\sum_{i,j} p(i,j)\log\frac{p(i,j)}{s(i)x^*(j)}$. By the low of total probability, $p(i,j)$ is rewritten as $\sum_h p(i|j,h)p(h|j)x^*(j)$, where $p(i|j,h) = p(i|h) = A(i,h)$ because $s(i)$ is completely defined by $x(h)$. In addition, we call $F(h,j) = p(h|j)$, from which it follows that $X = FX^*$, where F is also a stochastic matrix. Therefore, $I(S;X^*)$ is given by

$$I(S;X^*) = E\left[\sum_{i,j} A(i,\cdot)F(\cdot,j)X^*(j)\log\frac{A(i,\cdot)F(\cdot,j)}{S(i)}\right] \tag{4.10}$$

Learning F. F is learned from training data. Specifically, let's indicate with $\bar{X} = [x_1,\cdots,x_N]$ and $\bar{X}^* = [x_1^*,\cdots,x_N^*]$ the training data points corresponding to the main and privileged domains, then F is learned by solving the following constrained optimization problem: $\min_F D_{KL}(\bar{X}\|F\bar{X}^*)$ s.t. F is a stochastic matrix with normalized columns. We compute F with Algorithm 2, which is a projected gradient method [187] with Armijo's condition

$$\begin{aligned} &D_{KL}(\bar{X}\|F^{k+1}\bar{X}^*) - D_{KL}(\bar{X}\|F^k\bar{X}^*) \\ &\qquad \leq \nu\langle\nabla_F D_{KL}(\bar{X}\|F^k\bar{X}^*), F^{k+1} - F^k\rangle \end{aligned} \tag{4.11}$$

where k is the iteration index. The computation of $\nabla_F D_{KL}(\bar{X}\|F\bar{X}^*)$ is fairly simple, and can be found as a special case in [188].

Missing auxiliary views. Training samples with missing auxiliary view affect only $I(S;X^*)$. The issue is seamlessly handled by estimating F and the average in (4.10) by using only the samples that have the auxiliary view.

Algorithm 4 Projected gradient minimization for Q_f or Q_g

Chose $0 < \eta < 1$, $0 < \nu < 1$.
Initialize A^1 for Q_g (or B^1 for Q_f). Set $\rho = 1$.
for $k = 1, 2, \cdots$ **do**
  if ρ satisfies (4.16) for Q_g (or (4.17) for Q_f) **then**
    Repeatedly increase it by $\rho \leftarrow \rho/\eta$ until either ρ does not satisfy (4.16) (or (4.17)) or $A(\rho/\eta) = A(\rho)$ (or $B(\rho/\eta) = B(\rho)$)
  else
    Repeatedly decrease ρ by $\rho \leftarrow \rho/\eta$ until ρ satisfies (4.16) (or (4.17))
  end if
  Set $A^{k+1} = \max\{0, A^k - \rho\nabla_A Q_g(A^k, B)\}$
  (Set $B^{k+1} = \max\{0, B^k - \rho\nabla_B Q_f(B^k, A)\}$)
  Normalize to 1 the columns of A^{k+1} (or B^{k+1})
end for

	LB-LMIBPI	SVM	SVM-R	RankTr	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI	UB-LMIBPI	SVM2k	KCCA
brush hair	78.49 ± 4.99	78.50 ± 6.68	79.67 ± 4.25	78.16 ± 4.40	79.17 ± 6.40	77.50 ± 5.78	77.00 ± 6.79	**80.66 ± 4.85**	84.48 ± 5.03	85.00 ± 5.93	78.00 ± 8.08
dive	78.83 ± 4.23	80.66 ± 3.00	73.63 ± 4.11	79.66 ± 7.97	79.83 ± 3.64	82.50 ± 2.63	83.00 ± 2.33	**83.16 ± 5.23**	90.79 ± 5.94	87.16 ± 3.14	85.00 ± 2.07
drink	69.37 ± 6.62	69.16 ± 6.00	68.5 ± 5.43	**75.50 ± 6.18**	69.17 ± 5.80	69.50 ± 6.13	68.50 ± 5.41	72.36 ± 5.83	81.98 ± 7.48	74.33 ± 7.16	69.83 ± 6.20
eat	67.04 ± 5.86	67.66 ± 4.00	69.63 ± 5.00	**75.08 ± 2.10**	71.00 ± 5.16	71.50 ± 6.35	67.50 ± 6.58	74.85 ± 4.61	81.98 ± 4.95	76.00 ± 6.62	69.00 ± 6.19
golf	78.83 ± 3.68	81.50 ± 4.11	70.43 ± 4.02	74.66 ± 3.01	80.67 ± 7.25	80.00 ± 6.23	78.66 ± 4.76	**85.47 ± 5.66**	90.45 ± 3.68	85.33 ± 6.92	78.16 ± 5.29
hug	80.66 ± 3.35	81.83 ± 4.18	80.43 ± 4.66	82.82 ± 3.90	81.50 ± 3.72	83.33 ± 5.03	81.33 ± 4.83	**85.97 ± 3.44**	91.11 ± 5.61	87.83 ± 3.93	82.66 ± 4.91
jump	74.16 ± 3.70	74.16 ± 4.30	69.90 ± 2.87	75.66 ± 2.81	76.33 ± 6.42	77.00 ± 6.17	76.16 ± 6.03	**79.83 ± 2.65**	84.52 ± 3.62	81.16 ± 4.90	78.33 ± 7.37
pick	65.59 ± 3.88	63.83 ± 5.42	67.53 ± 3.67	78.33 ± 3.56	66.83 ± 4.19	65.16 ± 5.52	63.83 ± 5.15	**79.83 ± 3.56**	85.42 ± 3.79	68.50 ± 3.88	64.33 ± 6.09
punch	83.76 ± 3.20	83.66 ± 4.10	81.20 ± 5.72	87.33 ± 4.90	84.33 ± 5.89	84.83 ± 4.47	83.33 ± 4.37	**92.06 ± 3.54**	95.38 ± 3.54	86.16 ± 4.30	83.83 ± 4.23
sit	75.33 ± 3.10	75.16 ± 4.83	71.66 ± 9.8	76.33 ± 3.67	74.00 ± 4.60	75.16 ± 5.95	73.50 ± 5.05	**76.99 ± 3.58**	85.29 ± 4.05	77.50 ± 5.78	75.33 ± 5.92

Table 4.1: HMDB dataset. Classification accuracies for one-vs-all binary classifications. The HOF features represent the main view, and the HOG features the auxiliary view. Best accuracies are highlighted in boldface.

Optimization

When A is known, (4.8) is a soft-margin SVM problem. Instead, when the SVM parameters are known, (4.8) becomes

$$\min_A \; I(S;X) - \gamma I(X^*;S) + \frac{C}{N}\sum_{i=1}^{N}\xi_i \qquad (4.12)$$
$$\text{s.t.} \quad \xi_i = \max\{0, 1 - y_i(\langle w, \phi(x_i, A)\rangle + b)\} \; .$$

Since the soft-margin problem is convex, if also (4.12) is convex, then an alternating direction method is guaranteed to converge. In general, the mutual informations in (4.12) are convex functions of $q(S|X)$ [186]. The last term is also convex, however, the constraints define a non-convex set due to the discontinuity of the hinge loss function. Smoothing the hinge loss turns (4.12) into a convex problem, and allows to use an alternating direction method with variable splitting combined with the augmented Lagrangian method. This is done by setting $f(A) = I(S;X) - \gamma I(X^*;S)$, $g(B) = \frac{C}{N}\sum_{i=1}^{N}\xi_i$, and then solving $\min_A\{f(A) + g(B) : A - B = 0\}$.

For smoothing the hinge loss we use the Nesterov smoothing technique [189], used also in [190], which requires choosing a proximal function, and then computing the smoothed slack variables in

	LB-LMIBPI	SVM	SVM-R	RankTr	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI	UB-LMIBPI	SVM2k	KCCA
brush hair	91.67 ± 3.42	90.66 ± 3.06	73.33 ± 4.15	82.33 ± 4.09	**93.66 ± 1.72**	85.33 ± 5.56	90.33 ± 3.66	93.50 ± 2.42	93.83 ± 1.77	88.33 ± 9.52	91.33 ± 3.99
dive	88.50 ± 4.19	81.16 ± 4.44	83.83 ± 4.84	82.50 ± 3.16	84.50 ± 3.68	79.00 ± 5.73	83.00 ± 4.14	**86.67 ± 2.61**	90.83 ± 3.26	83.50 ± 3.96	83.50 ± 2.65
drink	76.83 ± 4.26	76.66 ± 6.52	68.83 ± 8.99	71.16 ± 8.01	75.16 ± 7.13	70.50 ± 3.33	77.16 ± 7.37	**78.83 ± 5.21**	81.00 ± 4.92	77.16 ± 5.44	79.16 ± 6.09
eat	75.00 ± 5.27	73.00 ± 5.70	72.83 ± 5.33	73.50 ± 5.05	75.50 ± 5.82	70.50 ± 4.23	74.33 ± 5.94	**78.00 ± 4.07**	79.33 ± 3.16	76.83 ± 8.02	77.33 ± 5.56
golf	84.50 ± 2.36	82.33 ± 3.61	78.83 ± 4.84	85.33 ± 5.43	86.16 ± 4.16	77.16 ± 6.18	84.33 ± 3.70	**88.17 ± 4.87**	89.83 ± 5.00	88.00 ± 3.02	89.00 ± 3.35
hug	86.00 ± 4.79	83.83 ± 4.84	74.83 ± 4.87	**87.33 ± 3.70**	86.33 ± 3.49	80.33 ± 6.17	83.83 ± 4.01	86.83 ± 4.34	87.50 ± 5.05	85.50 ± 4.23	86.16 ± 3.77
jump	80.50 ± 4.85	77.33 ± 7.78	78.33 ± 4.08	75.50 ± 4.30	79.66 ± 4.21	69.50 ± 5.62	**80.83 ± 6.24**	80.67 ± 4.98	84.33 ± 7.17	77.50 ± 4.46	81.50 ± 6.95
pick	73.17 ± 5.47	70.16 ± 11.1	65.83 ± 5.22	68.33 ± 3.33	72.33 ± 4.45	60.16 ± 7.09	70.66 ± 5.10	**74.83 ± 5.00**	75.17 ± 7.47	72.83 ± 4.44	73.50 ± 5.95
punch	88.10 ± 2.95	84.00 ± 5.67	83.83 ± 4.90	84.50 ± 4.44	**88.16 ± 5.58**	79.33 ± 4.66	86.66 ± 4.90	87.67 ± 4.10	88.83 ± 4.38	82.00 ± 5.76	82.33 ± 5.67
sit	78.00 ± 4.76	74.16 ± 6.15	76.50 ± 6.82	75.33 ± 4.14	76.33 ± 6.02	69.33 ± 7.70	76.33 ± 5.14	**82.00 ± 5.58**	83.17 ± 5.58	73.50 ± 6.82	75.33 ± 6.22

Table 4.2: HMDB dataset - HIK. Classification accuracies for one-vs-all binary classifications. The HOF features represented main data, and HOG features auxiliary data. Best accuracies are highlighted in boldface.

this way $\xi_{i,\sigma} = \max_{0\leq u_i\leq 1} u_i(1 - y_i w^\top A x_i) - \frac{\sigma}{2}\|w x_i^\top\|_\infty u_i^2$, which gives

$$\xi_{i,\sigma} = \begin{cases} 0 & y_i w^\top A x_i > 1\,, \\ (1 - y_i w^\top A x_i) - \frac{\sigma}{2}\|w x_i^\top\|_\infty & y_i w^\top A x_i < 1 \\ & -\sigma\|w x_i^\top\|_\infty, \\ \frac{(1-y_i w^\top A x_i)^2}{2\sigma\|w x_i^\top\|_\infty} & \text{otherwise.} \end{cases} \tag{4.13}$$

where σ is a smoothing parameter. In this way, the minimization can be carried out with the Fast Alternating Linearization Method (FALM) [191]. This allows simpler computations, and has performance guarantees when ∇f and ∇g are Lipschitz continuous, which is the case, given the smoothing technique that we used.

FALM splits the minimization of the augmented Lagrangian function into two simpler functions to be minimized alternatively, which are given by

$$Q_g(A, B) = f(A) + g(B) + \langle \nabla g(B), A - B\rangle + \frac{1}{\mu_g} D_{KL}(A||B) \tag{4.14}$$

$$Q_f(B, A) = f(A) + g(B) + \langle \nabla f(A), B - A\rangle + \frac{1}{\mu_g} D_{KL}(B||A) \tag{4.15}$$

The FALM iteration is given in Algorithm 3. Since A is a stochastic matrix, the KL-divergence regularization is used in place of the squared Frobenius norm.

Note that lines 3 and 4 of Algorithm 3 are constrained optimizations, requiring A and B to be stochastic matrices with normalized columns. They are implemented by Algorithm 4, a projected gradient method [187] with Armijo's rule that for Q_g and Q_f is given by

$$Q_g(A^{k+1}, B) - Q_g(A^k, B) \leq \nu\langle \nabla_A Q_g(A^k, B), A^{k+1} - A^k\rangle \tag{4.16}$$

	LB-LMIBPI	SVM	SVM-R	RankTr	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI	UB-LMIBPI	SVM2k	KCCA
Thunder snake	56.84 ± 3.21	57.09 ± 3.14	52.42 ± 2.53	57.88 ± 3.57	**60.17 ± 2.29**	59.12 ± 2.00	56.79 ± 2.33	59.70 ± 2.62	63.31 ± 3.23	61.70 ± 1.35	57.05 ± 2.39
Ringneck snake	62.03 ± 2.62	63.31 ± 2.76	53.55 ± 3.78	62.25 ± 1.46	63.14 ± 2.45	63.93 ± 2.82	63.47 ± 2.03	**64.72 ± 2.36**	68.36 ± 2.65	67.49± 2.43	64.46 ± 2.55
Hognose snake	57.71 ± 1.56	60.11 ± 1.34	55.74 ± 2.36	55.33 ± 2.77	59.10 ± 1.81	59.73 ± 2.10	60.55 ± 0.96	**60.63 ± 1.58**	65.32 ± 4.25	61.53± 1.36	60.03 ± 1.32
Green snake	68.11 ± 2.85	71.46 ± 1.39	55.43 ± 6.54	62.20 ± 2.99	70.66 ± 1.83	71.12 ± 1.40	70.03 ± 2.20	**72.72 ± 2.17**	77.82 ± 5.36	72.64± 1.45	68.80 ± 2.59
King snake	62.75 ± 1.57	60.20 ± 1.83	61.14 ± 3.53	59.70 ± 3.74	**63.84 ± 2.07**	59.81 ± 1.54	59.92 ± 2.01	61.70 ± 4.96	65.38 ± 2.36	64.89± 1.41	60.70 ± 1.78
Garter snake	66.72 ± 5.23	69.02 ± 3.25	57.07 ± 4.79	66.47 ± 2.58	66.02 ± 2.34	69.17 ± 2.90	**69.21 ± 2.86**	68.23 ± 1.79	70.23 ± 5.36	72.97± 2.37	69.65 ± 3.31
Water snake	70.26 ± 1.47	71.94 ± 1.91	64.80 ± 9.72	67.86 ± 3.40	68.82 ± 2.86	72.21 ± 1.83	71.34 ± 1.95	**72.72 ± 4.96**	73.21 ± 3.42	72.50± 2.13	70.11 ± 1.54
Vine snake	67.85 ± 3.52	78.92 ± 2.04	73.04 ± 5.00	69.97 ± 4.87	74.86 ± 2.09	**79.05 ± 2.14**	78.45 ± 1.95	77.91 ± 1.77	78.92 ± 4.02	80.15± 2.58	78.45 ± 1.72
Night snake	52.42 ± 6.32	53.97 ± 3.62	54.01 ± 2.25	52.96 ± 3.23	55.19 ± 1.76	55.26 ± 3.39	55.00 ± 3.44	**57.09 ± 2.14**	60.17 ± 4.23	55.48± 3.35	54.51 ± 3.32
Boa constrictor	61.90 ± 2.41	61.76 ± 1.87	59.03 ± 4.80	59.64 ± 3.40	62.66 ± 1.23	**62.92 ± 1.65**	61.60 ± 2.08	60.86 ± 1.72	63.76 ± 6.03	63.46± 2.10	61.19 ± 1.48
Rock python	57.88 ± 6.85	60.39 ± 2.36	56.84 ± 2.83	57.71 ± 2.59	58.43 ± 2.92	60.14 ± 2.46	59.50 ± 1.44	**60.59 ± 1.54**	61.14 ± 2.36	60.92± 2.16	58.78 ± 1.81
Indian cobra	61.90 ± 3.56	65.21 ± 2.84	59.04 ± 6.87	63.76 ± 3.92	65.88 ± 2.55	**66.88 ± 2.83**	64.92 ± 3.55	63.19 ± 2.97	64.53 ± 4.02	68.80± 1.52	65.42 ± 3.39
Green mamba	65.24 ± 1.69	68.50 ± 2.33	62.71 ± 9.11	66.77 ± 5.23	67.56 ± 2.39	68.36 ± 1.95	67.09 ± 2.47	**68.72 ± 3.50**	71.46 ± 5.36	70.14± 1.67	67.27 ± 2.16
Sea snake	72.72 ± 2.56	77.42 ± 1.64	68.36 ± 4.46	77.22 ± 2.03	76.17 ± 1.61	**77.55 ± 1.84**	77.10 ± 1.54	77.22 ± 1.70	78.53 ± 2.39	78.51± 1.41	73.06 ± 1.66
Horned viper	67.90 ± 1.49	69.92 ± 1.25	63.95 ± 5.17	67.74 ± 2.75	67.41 ± 1.94	69.49 ± 2.39	69.54 ± 1.01	**71.46 ± 3.21**	74.86 ± 6.45	71.80± 2.72	68.98 ± 1.31
Diamondback	64.27 ± 2.53	66.18 ± 2.56	61.90 ± 3.09	64.07 ± 2.60	63.48 ± 3.12	66.60 ± 1.81	65.89 ± 2.12	**69.92 ± 2.42**	71.94 ± 5.01	69.32± 1.59	65.69 ± 2.60
Sidewinder	67.41 ± 3.92	69.55 ± 2.34	60.28 ± 4.82	68.11 ± 3.21	67.85 ± 2.64	68.93 ± 5.54	68.70 ± 2.50	**70.66 ± 3.19**	72.77 ± 4.03	69.88± 7.25	66.46 ± 1.84

Table 4.3: ImageNet dataset. Classification accuracies for one-vs-all binary classifications. The BoW from the whole image is the main view, and the BoW from the bounding box region is the auxiliary view. Best accuracies are highlighted in boldface.

	LB-LMIBPI	SVM	SVM-R	RankTr	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI	UB-LMIBPI	SVM2k	KCCA
Thunder snake	54.32 ± 2.21	58.21 ± 1.95	55.35 ± 3.45	**62.31 ± 3.21**	61.59 ± 1.63	56.74 ± 3.09	59.71 ± 2.15	60.52 ± 3.21	64.73 ± 3.21	59.43 ± 2.49	58.40 ± 2.75
Ringneck snake	65.23 ± 1.78	64.60 ± 2.57	56.84 ± 4.02	64.32 ± 2.15	65.49 ± 2.86	61.26 ± 2.67	61.93 ± 1.53	**67.51 ± 2.11**	68.32 ± 2.56	62.88 ± 2.96	62.80 ± 2.47
Hognose snake	58.43 ± 2.21	60.00 ± 1.38	59.14 ± 2.65	61.22 ± 1.83	59.52 ± 2.29	54.78 ± 2.72	57.47 ± 3.01	**63.72 ± 1.89**	66.51 ± 3.45	55.90 ± 2.70	59.25 ± 1.93
Green snake	69.25 ± 1.32	71.59 ± 1.65	60.66 ± 4.20	72.23 ± 2.68	70.34 ± 1.21	62.42 ± 7.44	65.03 ± 1.43	**73.21 ± 2.63**	75.42 ± 2.87	66.97 ± 7.13	70.46 ± 1.80
King snake	64.41 ± 3.22	63.46 ± 1.48	53.00 ± 2.74	66.21 ± 3.25	65.76 ± 1.61	55.05 ± 4.50	60.74 ± 1.32	**67.31 ± 3.25**	68.22 ± 5.36	56.86 ± 5.81	61.72 ± 2.60
Garter snake	71.22 ± 2.33	69.27 ± 2.52	60.17 ± 4.26	**72.33 ± 3.36**	70.34 ± 2.37	65.29 ± 3.59	64.40 ± 1.86	71.44 ± 4.15	73.25 ± 4.32	68.88 ± 2.13	68.49 ± 1.32
Water snake	71.31 ± 3.25	72.32 ± 1.84	62.75 ± 2.98	73.45 ± 4.51	71.88 ± 1.23	69.65 ± 1.85	65.98 ± 1.64	**74.41 ± 2.63**	76.85 ± 3.25	69.76 ± 1.74	70.05 ± 1.68
Vine snake	79.05 ± 2.51	79.12 ± 2.27	67.64 ± 5.86	78.32 ± 3.72	79.05 ± 1.96	76.21 ± 4.15	73.67 ± 1.25	**80.11 ± 3.65**	83.22 ± 2.31	77.55 ± 2.97	78.13 ± 1.74
Night snake	56.33 ± 3.11	55.69 ± 3.22	52.90 ± 2.10	**59.23 ± 2.23**	57.27 ± 1.74	54.60 ± 2.96	55.91 ± 3.08	58.32 ± 4.11	60.11 ± 4.28	55.65 ± 2.93	55.43 ± 2.57
Boa constrictor	64.80 ± 2.11	64.69 ± 1.78	52.61 ± 2.84	63.21 ± 1.80	65.26 ± 1.81	57.18 ± 5.62	63.00 ± 1.07	**66.21 ± 2.56**	68.42 ± 3.35	60.04 ± 6.85	64.52 ± 2.45
Rock python	62.22 ± 1.78	61.99 ± 1.63	52.86 ± 2.05	61.55 ± 2.35	60.57 ± 1.75	54.07 ± 4.34	58.53 ± 1.76	**64.32 ± 3.89**	68.23 ± 2.36	55.87 ± 4.91	60.59 ± 2.38
Indian cobra	66.51 ± 1.96	67.23 ± 2.33	59.73 ± 5.50	67.23 ± 4.36	**68.57 ± 1.95**	62.42 ± 4.82	65.19 ± 2.14	67.89 ± 2.75	71.08 ± 4.83	64.23 ± 3.52	65.98 ± 2.73
Green mamba	69.22 ± 1.32	68.18 ± 1.97	60.58 ± 4.23	71.22 ± 1.25	69.59 ± 1.68	64.79 ± 5.25	64.79 ± 2.03	**74.32 ± 1.56**	78.31 ± 3.96	65.82 ± 4.55	67.50 ± 1.86
Sea snake	78.53 ± 3.21	77.90 ± 9.05	69.07 ± 7.32	79.22 ± 3.38	81.48 ± 1.44	63.47 ± 9.81	75.70 ± 1.08	**82.32 ± 3.18**	83.41 ± 3.57	65.59 ± 9.21	79.07 ± 1.03
Horned viper	71.32 ± 4.51	70.64 ± 1.65	59.23 ± 4.16	71.23 ± 2.68	70.84 ± 1.29	60.01 ± 6.64	66.00 ± 2.21	**73.45 ± 3.75**	76.77 ± 2.58	64.05 ± 7.02	69.29 ± 1.79
Diamondback	68.31 ± 2.11	67.45 ± 2.09	58.00 ± 5.02	69.35 ± 3.65	68.55 ± 2.54	58.48 ± 5.65	63.44 ± 2.56	**71.22 ± 2.86**	72.31 ± 1.36	62.14 ± 4.87	65.32 ± 2.20
Sidewinder	70.21 ± 1.17	69.62 ± 2.10	63.23 ± 6.20	69.58 ± 4.11	71.39 ± 2.62	66.79 ± 3.49	67.54 ± 2.70	**72.36 ± 1.98**	78.33 ± 2.35	68.18 ± 2.34	69.10 ± 2.83

Table 4.4: ImageNet dataset - HIK. Classification accuracies for one-vs-all binary classifications. The BoW from the whole image represented main data, and the BoW from the bounding box region auxiliary data. Best accuracies are highlighted in boldface.

$$Q_f(B^{k+1}, A) - Q_f(B^k, A) \leq \nu \langle \nabla_B Q_f(B^k, A), B^{k+1} - B^k \rangle \tag{4.17}$$

where k is the iteration index. From (4.13), (4.9), (4.10) it is straightforward to compute $\nabla_A Q_g$, and $\nabla_B Q_f$. We leave those expressions out due to the limited space.

4.5 LMBPI bounds

4.5.1 Lower-bound LMIBPI

In addition to the proposed LMIBPI, we also deploy two more approaches, representing the upper and lower bounds of LMIBPI. The lower bound corresponds to eliminating the use of auxiliary information from LMIBPI. Computationally, this can be achieved very easily by setting $\gamma = 0$ in (4.8). For each of the experiments we have added a column indicated with `LB-LMIBPI`, which stands for lower-bound LMIBPI, and which represents this case.

4.5.2 Upper-bound LMIBPI

The upper-bound model for LMIBPI corresponds to the case for when main and auxiliary data are available at both training and testing time. We model this situation as in Figure 4.3. In particular, we allow the auxiliary data X^* to be compressed, and obtain S^*, as indicated by G_{in} in Figure 4.3. The desired output is identified by G_{out}, where S^* d-separates X^* and Y, and S d-separates X and Y. Also, we have that (S, S^*) d-separates (X, X^*) from Y. This means that we would like to have at the same time $I(X;Y|S) = 0$, $I(X^*;Y|S^*) = 0$, and $I(X, X^*;Y|S, S^*) = 0$. So, we should compress X and X^* as much as possible, provided that S and S^* retain all the information about Y.

The multi-information of G_{in} and G_{out} of Figure 4.3 is given by

$$\mathcal{I}^{G_{in}} = I(S;X) + I(S^*;X^*) + I(Y;X,X^*) \,, \tag{4.18}$$

$$\mathcal{I}^{G_{out}} = I(S;X) + I(S^*;X^*) + I(S,S^*;Y) \,. \tag{4.19}$$

Since $I(Y;X,X^*)$ is constant, the functional for learning the optimal representation for S and S^* becomes

$$\mathcal{L}[q(S,S^*|X,X^*)] = I(S;X) + I(S^*,X^*) - \gamma I(S,S^*;Y) \,. \tag{4.20}$$

From (4.20) we derive the large-margin formulation of the upper-bound of LMIBPI. In particular, we restrict the search space for $q(S,S^*|X,X^*)$ by assuming that

$$S = \phi(X;A) \,, \tag{4.21}$$

$$S^* = \phi(X^*;A^*) \,, \tag{4.22}$$

where A and A^* are two suitable sets of parameters. Moreover, from S and S^* we aim at predicting the relevant information Y through the decision function given by

$$Y = \text{sign}\left(\left\langle w, \begin{bmatrix} S \\ S^* \end{bmatrix}\right\rangle + b\right) \,. \tag{4.23}$$

Therefore, the large-margin problem that we solve is the following

$$\begin{aligned} \min_{A,A^*,w,b,\xi_i} \; & I(S;X) + I(S^*;X^*) + \frac{\beta}{2}\|w\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i\left(\left\langle w, \begin{bmatrix} \phi(x_i, A) \\ \phi(x_i^*, A^*) \end{bmatrix} \right\rangle + b\right) \geq 1 - \xi_i \ , \\ & \xi_i \geq 0 \ , \quad \forall i \in \{1, \cdots, N\} \ . \end{aligned} \tag{4.24}$$

The optimization of problem (4.24), which we refer to as *upper-bound LMIBPI (UP-LMIBPI)* can be carried out with techniques similar to LMIBPI. In particular, we assume the relationships $S = AX$, and $S^* = A^*X^*$, where A and A^* are stochastic matrices with normalized columns. Although now we have to estimate A and A^*, the parameter γ of (4.8) has disappeared. The implementation of the optimization entails alternating between solving an SVM problem when A and A^* are known, and then keep every parameter fixed and update A, and subsequently update A^*. The update of A and A^* requires a set of equations similar to LMIBPI.

In the rest of this material we refer to the implementation of UP-LMIBPI as `UP-LMIBPI`, and for each experiment we report results corresponding to this case, where auxiliary information is available at testing time too (which actually makes it no-longer privileged!).

4.6 Experiments

We have performed experiments with four different datasets. With each dataset we train and test the following binary classifiers.

Single-view classifiers: Using only the main view, we train the SVM-Light [192] (indicated as `SVM`), the SVM-Rank [193] (indicated as `SVM-R`), and LMIBPI where we eliminate the use of auxiliary information by setting $\gamma = 0$ (indicated as `LB-LMIBPI`).

LUPI classifiers: We train the SVM+ [56] (indicated as `SVM+`, implemented by [194]), the Rank Transfer [10] (indicated as `RankTr`, and reimplemented by us), and our LMIBPI approach (indicated as `LMIBPI`). We also train the SVM2k [73] and test only the SVM that uses the main view (indicated as `SVM2k-LUPI`), and we perform kernel CCA (KCCA) [195] between main and auxiliary views, map the main view in feature space and train an SVM (indicated as `KCCA-LUPI`).

Two-view classifiers: Using main and auxiliary views, we train the SVM2k (indicated as `SVM2k`),

and we also use KCCA between views to map them in feature space, train two SVMs and average the outputs (indicated as `KCCA`). Finally, we also extend LMIBPI (details are omitted for lack of space) to fuse main and auxiliary views (indicated as `UB-LMIBPI`). Note that for these classifiers main and auxiliary views are used also during testing. So, their performances represent the upper bound for the corresponding LUPI versions.

Model selection: We use the same joint cross validation and model selection procedure described in [10], based on 5-fold cross-validation to select the best parameters and use them to retrain on the complete set. The main parameters to select are C, β, γ, and m, the number of columns of A. The C's and β's were searched in the range $\{10^{-3}, \cdots, 10^{3}\}$, the γ's in the range $\{0.1, 0.3, 0.5\}$, and the m's in the range $\{50, 70, 90\}$.

Performance: For each binary classification experiment we randomly select the same number of positive and negative samples for training, and the same for testing. Each experiment is repeated 10 times and average classification accuracy and standard deviation are reported.

Kernels: We use the linear and the histogram intersection (HIK). Due to space constraints we report table results for the linear case, and include figures for the HIK. Tables for more non-linear kernels are omitted for lack of space.

HMDB dataset: The HMDB dataset [196] is a video dataset for action recognition, composed of 51 classes. Each class has approximately 100 videos. We have randomly selected 10 classes, and we have considered the binary classification between one class versus the rest. With this experiment we test whether computing an auxiliary feature only during training, can be used to improve the recognition during testing. This would mean a performance improvement while saving computing power. For every video we extracted two bag-of-word (BoW) representations, one given by HOF descriptors, and one by HOG descriptors. We used dictionaries of size 400, learned with VLFeat [197]. We used 70 samples per class for training and 30 for testing. The HOF descriptors were set to the main view, and the HOG's represented auxiliary information. Table 4.1 collects the classification accuracies for the linear kernel. As expected, LUPI classifiers improve upon single-view, and `LMIBPI` outperforms the others 8 out of 10 times in the linear case, and 6 times with HIK. See Figure 4.4 (top row).

Table 4.2 provides the classification accuracy for the same experiment reported in Table 4.1, with the exception that the linear kernel here is replaced by the HIK kernel. Figure 4.5 shows the

differences between the accuracies of `LMIBPI` versus `RankTr`, `SVM+`, `SVM2k-LUPI` and `KCCA-LUPI`. These plots highlight that `LMIBPI` compares favorably. Figure 4.6 shows how `LMIBPI` compares against `LB-LMIBPI` and `UB-LMIBPI`. If we consider the *performance gap* between lower and upper bound accuracies, and we identify with 0% the lower bound, and with 100% the upper bound, on average, using the auxiliary information allows recovering 47.1% of the performance gap in the linear kernel case, and 35.4% when HIK is used. Also, using HIK improves the average performance from $81.11 \pm 5.80\%$ to $83.71 \pm 5.75\%$.

Time complexity: `LMIBPI` estimates F only once, and then iterates between optimizing A and a SVM. Both components are fast, also thanks to the derivation in the primal space. In addition, Figure 4.7 shows the accuracy convergence for the `drink` class of the HMDB dataset for different m's. We observed that less than 10 iterations were enough to reach convergence most of the time.

ImageNet dataset: We use the ImageNet [48] object categories of the 2012 challenge, also used in [10]. This subset has bounding box annotations, and we test whether they can improve recognition when used as auxiliary information. We use the group of snakes, which has 17 classes, for a total of 7746 images (some bounding boxes did not have images). For each sample we extracted a BoW from the entire image to be used as main view, and a BoW from the image portion in the bounding box to be used as auxiliary view. The descriptor used was dense SIFT [197] with a vocabulary size of 400. The classification task is between one snake class versus all the others. We use 200 samples per class for training and the rest for testing. Table 4.3 summarizes the classification accuracy results. Even here LUPI classifiers improve upon single-view, and `LMIBPI` outperforms the others 10 out of 17 times in the linear case, and 13 times with HIK. See Figure 4.4 (middle row).

Table 4.4 provides the classification accuracy for the same experiment reported in Table 4.3, with the exception that the linear kernel here is replaced by the HIK kernel. Figure 4.8 shows the differences between the accuracies of `LMIBPI` versus `RankTr`, `SVM+`, `SVM2k-LUPI` and `KCCA-LUPI`. These plots highlight that `LMIBPI` compares favorably. Figure 4.9 shows how `LMIBPI` compares against `LB-LMIBPI` and `UB-LMIBPI`. On average, using the auxiliary information allows recovering 47.4% of the performance gap in the linear kernel case, and 50.6% when HIK is used. Also, using HIK improves the average performance from $66.94 \pm 6.39\%$ to $69.92 \pm 6.39\%$.

CGD2011 dataset: The CGD2011 dataset [198] contains 20 gesture classes, each of which has

	LB-LMIBPI	SVM	SVM-R	RankTr	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI	UB-LMIBPI	SVM2k	KCCA
vieniqui	52.89 ± 1.98	49.72 ± 4.96	52.83 ± 6.24	52.11 ± 4.83	50.27 ± 4.20	51.16 ± 4.50	48.33 ± 4.84	**54.00 ± 1.32**	56.50 ± 5.06	50.72 ± 2.99	52.66 ± 3.38
prendere	53.95 ± 4.23	52.39 ± 3.00	56.38 ± 4.10	54.50 ± 3.87	**58.05 ± 2.34**	54.83 ± 3.16	52.44 ± 3.34	57.28 ± 4.18	61.61 ± 8.32	56.50± 3.54	57.50 ± 2.72
sonostufo	55.95 ± 2.56	52.27 ± 3.97	57.00 ± 4.61	57.11 ± 4.19	**59.44 ± 3.74**	54.05 ± 4.29	51.33 ± 3.56	59.28 ± 2.19	66.33 ± 7.25	58.88± 4.47	57.00 ± 4.55
chevuoi	60.00 ± 5.62	57.55 ± 4.16	57.72 ± 5.12	55.22 ± 3.16	54.77 ± 4.24	59.22 ± 4.22	57.00 ± 3.45	**61.11 ± 2.84**	65.83 ± 6.10	67.05± 2.12	61.11 ± 2.27
daccordo	61.53 ± 5.25	65.83 ± 3.34	**67.00 ± 3.87**	63.61 ± 2.34	65.50 ± 5.28	**67.00 ± 3.59**	63.27 ± 3.40	64.86 ± 3.94	67.33 ± 8.25	74.83± 3.54	65.66 ± 4.69
perfetto	66.61 ± 4.05	64.55 ± 4.54	62.05 ± 3.46	60.11 ± 4.60	64.16 ± 2.40	64.94 ± 4.79	65.83 ± 4.26	**67.72 ± 7.71**	68.11 ± 7.23	64.05± 3.78	66.16 ± 5.14
vattene	61.83 ± 7.02	65.66 ± 3.19	62.27 ± 2.16	61.83 ± 5.43	64.55 ± 4.07	**65.72 ± 1.88**	63.88 ± 3.24	65.11 ± 5.07	68.70 ± 4.21	67.44± 3.47	66.83 ± 2.74
basta	67.00 ± 6.56	65.11 ± 5.18	65.44 ± 3.35	63.38 ± 4.37	64.11 ± 2.55	65.27 ± 3.91	62.72 ± 5.42	**68.11 ± 4.28**	69.22 ± 6.32	74.94± 6.21	72.11 ± 4.87
buonissimo	56.56 ± 8.02	52.44 ± 12.1	**58.64 ± 6.57**	58.55 ± 5.18	55.94 ± 5.17	56.05 ± 5.82	54.50 ± 4.52	57.67 ± 5.71	61.50 ± 2.35	65.38± 6.76	55.11 ± 5.00
cheduepalle	63.89 ± 2.01	66.27 ± 2.29	66.44 ± 2.82	65.83 ± 2.87	67.33 ± 3.33	66.66 ± 1.81	64.94 ± 2.47	**67.72 ± 2.01**	68.11 ± 3.05	76.05± 2.67	70.72 ± 2.85
cosatifarei	58.78 ± 6.20	61.99 ± 3.29	62.33 ± 4.03	61.50 ± 4.17	61.61 ± 4.40	**64.50 ± 3.55**	61.50 ± 5.25	62.11 ± 4.98	67.17 ± 6.24	64.88± 4.40	63.94 ± 5.75
fame	61.11 ± 5.23	59.55 ± 2.98	60.66 ± 2.87	61.38 ± 3.34	62.66 ± 3.90	**63.38 ± 3.47**	58.33 ± 1.50	60.55 ± 2.35	66.61 ± 7.22	65.94± 3.52	61.44 ± 4.40
noncenepiu	53.83 ± 1.99	52.61 ± 4.39	53.11 ± 3.55	53.83 ± 2.70	52.94 ± 3.21	**54.94 ± 4.71**	51.33 ± 3.73	**54.94 ± 3.01**	58.55 ± 5.23	55.83± 5.57	56.44 ± 4.21
furbo	63.39 ± 5.06	67.27 ± 3.56	65.22 ± 3.65	63.00 ± 3.10	66.33 ± 1.53	68.66 ± 3.30	66.05 ± 2.93	**68.70 ± 4.65**	72.22 ± 4.31	73.05± 1.87	70.22 ± 4.71
combinato	56.67 ± 7.26	56.33 ± 2.41	59.83 ± 3.73	58.55 ± 4.55	61.05 ± 3.38	58.83 ± 2.61	55.83 ± 2.43	**62.11 ± 2.26**	65.83 ± 6.32	75.00± 2.83	64.05 ± 3.66
freganiente	55.00 ± 4.25	52.38 ± 3.21	58.77 ± 3.28	56.94 ± 4.56	54.05 ± 6.20	56.94 ± 3.74	53.00 ± 3.12	**59.28 ± 4.89**	64.16 ± 3.95	58.05± 4.77	54.44 ± 4.59
seipazzo	60.61 ± 6.52	57.16 ± 4.58	55.50 ± 4.89	55.00 ± 3.92	53.55 ± 3.90	60.05 ± 3.23	58.05 ± 4.37	**61.72 ± 5.02**	65.44 ± 6.03	70.77± 2.97	61.94 ± 5.92
tantotempo	59.89 ± 5.15	61.50 ± 2.95	60.75 ± 3.75	59.27 ± 3.63	63.66 ± 1.96	62.22 ± 2.35	61.27 ± 2.75	**63.80 ± 3.72**	67.27 ± 3.25	70.83± 3.22	65.33 ± 2.74
messidaccordo	54.83 ± 1.34	53.49 ± 8.88	57.15 ± 4.47	**59.05 ± 4.67**	**59.05 ± 2.98**	58.44 ± 2.39	55.66 ± 3.43	55.94 ± 3.17	59.05 ± 5.23	54.50± 4.76	58.50 ± 4.92
ok	53.06 ± 2.98	51.83 ± 2.95	**56.50 ± 10.2**	53.44 ± 3.62	52.50 ± 2.78	53.88 ± 3.39	50.22 ± 2.79	56.39 ± 2.19	60.75 ± 6.35	51.27± 3.43	52.77 ± 3.16

Table 4.5: CGD2011 dataset. Classification accuracies for one-vs-all binary classifications. The HOF features are used as main view, and histograms of joint positions are used as auxiliary view. Best accuracies are highlighted in boldface.

	LB-LMIBPI	SVM	SVM-R	RankTr	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI	UB-LMIBPI	SVM2k	KCCA
vieniqui	55.78 ± 4.72	50.22 ± 2.88	53.94 ± 3.70	52.61 ± 3.91	52.22 ± 4.47	51.38 ± 5.89	54.38 ± 3.88	**60.27 ± 3.77**	62.27 ± 5.23	56.16 ± 3.97	55.44 ± 3.90
prendere	59.44 ± 2.53	56.05 ± 2.82	50.77 ± 1.36	54.39 ± 2.95	58.61 ± 2.91	48.27 ± 3.39	58.50 ± 2.23	**61.27 ± 1.83**	64.61 ± 3.25	58.05 ± 6.15	61.38 ± 4.21
sonostufo	60.39 ± 3.27	56.11 ± 2.60	55.27 ± 4.64	56.72 ± 4.21	59.88 ± 4.33	53.83 ± 6.30	58.66 ± 3.95	**62.94 ± 2.79**	63.73 ± 3.50	67.55 ± 5.16	63.00 ± 4.31
chevuoi	60.17 ± 3.69	58.27 ± 2.88	59.22 ± 3.37	58.11 ± 2.38	58.66 ± 3.03	56.66 ± 3.20	59.33 ± 3.05	**65.05 ± 3.56**	67.86 ± 3.65	66.66 ± 3.12	67.22 ± 4.32
daccordo	66.06 ± 4.42	67.05 ± 3.53	63.88 ± 5.33	61.83 ± 3.07	67.94 ± 2.52	66.05 ± 8.54	66.70 ± 2.18	**71.66 ± 2.84**	73.44 ± 5.72	77.50 ± 5.17	78.50 ± 2.45
perfetto	66.94 ± 2.42	66.55 ± 4.51	62.33 ± 4.16	62.05 ± 3.52	66.33 ± 2.74	63.33 ± 8.61	63.00 ± 3.32	**68.44 ± 2.85**	71.11 ± 2.82	68.16 ± 5.39	60.55 ± 3.16
vattene	67.61 ± 1.56	64.50 ± 2.80	61.72 ± 3.04	60.28 ± 2.67	64.61 ± 2.83	62.11 ± 3.27	63.77 ± 2.92	**69.05 ± 2.63**	72.33 ± 3.45	73.44 ± 3.87	67.88 ± 3.77
basta	67.06 ± 5.14	67.84 ± 4.47	63.66 ± 6.52	63.44 ± 6.11	68.05 ± 2.76	59.66 ± 6.03	69.05 ± 3.46	**69.33 ± 3.01**	73.33 ± 2.80	71.11 ± 5.73	79.94 ± 4.07
buonissimo	57.78 ± 4.23	55.05 ± 3.84	57.33 ± 3.10	54.55 ± 3.83	56.61 ± 4.33	54.66 ± 2.78	56.66 ± 3.71	**63.50 ± 3.77**	65.96 ± 3.63	65.38 ± 6.00	60.66 ± 5.35
cheduepalle	67.61 ± 2.85	66.72 ± 2.44	65.88 ± 2.11	61.94 ± 5.63	66.88 ± 3.64	62.00 ± 4.99	63.50 ± 4.00	**70.33 ± 2.40**	74.44 ± 1.88	75.66 ± 2.61	70.55 ± 3.96
cosatifarei	65.88 ± 4.12	63.55 ± 3.20	59.72 ± 4.12	59.00 ± 3.85	61.77 ± 3.30	58.11 ± 7.29	61.38 ± 4.38	**68.33 ± 3.37**	73.05 ± 3.36	67.11 ± 6.24	62.44 ± 4.80
fame	62.05 ± 5.25	56.68 ± 2.31	57.88 ± 2.97	58.83 ± 3.70	63.22 ± 4.87	52.94 ± 5.09	57.05 ± 2.62	**64.94 ± 1.86**	67.00 ± 5.71	66.22 ± 4.37	59.50 ± 4.22
noncenepiu	58.50 ± 4.32	53.22 ± 3.30	53.16 ± 1.83	55.61 ± 3.44	54.88 ± 4.22	53.50 ± 5.11	55.44 ± 3.74	**61.66 ± 3.26**	63.72 ± 3.88	60.83 ± 4.35	53.94 ± 2.36
furbo	63.44 ± 3.12	67.66 ± 3.27	63.33 ± 5.47	61.66 ± 3.29	65.88 ± 3.63	62.88 ± 4.97	63.55 ± 2.91	**68.55 ± 1.61**	72.22 ± 3.69	67.83 ± 3.35	60.88 ± 6.55
combinato	63.83 ± 2.12	59.44 ± 2.91	56.99 ± 2.54	59.78 ± 1.68	64.16 ± 1.92	59.83 ± 4.54	59.72 ± 1.98	**65.72 ± 4.19**	69.34 ± 2.32	78.16 ± 6.64	76.50 ± 1.65
freganiente	59.13 ± 3.25	51.94 ± 2.52	56.11 ± 3.22	54.89 ± 4.82	54.88 ± 3.52	52.05 ± 5.07	52.33 ± 2.34	**62.61 ± 2.61**	67.88 ± 3.23	59.16 ± 2.97	57.83 ± 2.54
seipazzo	60.56 ± 4.12	56.55 ± 4.11	56.77 ± 4.30	53.16 ± 6.42	55.05 ± 2.81	50.50 ± 5.74	57.44 ± 2.50	**63.55 ± 4.08**	65.96 ± 4.53	69.00 ± 5.28	64.88 ± 4.28
tantotempo	66.88 ± 4.15	65.72 ± 2.59	56.56 ± 3.61	59.11 ± 2.58	68.05 ± 1.94	63.55 ± 6.33	62.00 ± 2.99	**68.11 ± 2.38**	72.22 ± 4.78	75.38 ± 3.84	72.33 ± 4.03
messidaccordo	61.38 ± 3.80	60.66 ± 4.74	55.33 ± 2.64	58.33 ± 2.61	63.22 ± 4.10	58.61 ± 4.90	60.88 ± 4.67	**65.05 ± 2.72**	67.11 ± 3.21	59.27 ± 4.85	56.88 ± 6.03
ok	56.33 ± 1.22	53.15 ± 2.39	53.72 ± 2.49	52.55 ± 3.93	54.61 ± 3.56	50.66 ± 5.85	55.27 ± 3.18	**59.61 ± 3.43**	62.44 ± 2.65	56.27 ± 6.26	63.55 ± 2.64

Table 4.6: CGD2011 dataset - HIK. Classification accuracies for one-vs-all binary classifications. The HOF features represented main data, and histograms of joint positions were used as auxiliary data. Best accuracies are highlighted in boldface.

about 400 RGB-D videos, along with skeleton tracking data. Since skeleton tracking is typically more expensive to obtain, we test whether by using it as auxiliary data it can boost performance. We perform one-vs-all classification with 100 samples per class for training and 90 for testing. We used a BoW with dictionary size 100 based on HOF features as main view. For the auxiliary view, from a video we extract a histogram of the joint positions, accumulated over all the frames of the sequence. Specifically, at every frame we place a spatial grid aligned with the head position of an individual and bin the position of each of the joints with respect to the grid. The resulting count is normalized and produces a histogram with 100 bins (Figure 4.10). Table 4.5 shows the classification accuracies. The LUPI classifiers improve upon single-view, and `LMIBPI` outperforms the others 11 out of 20 times in the linear case, and all the times with HIK. See Figure 4.4 (bottom row).

Table 4.6 provides the classification accuracy for the same experiment reported in Table 4.5, with the exception that the linear kernel here is replaced by the HIK kernel. Figure 4.11 shows the differences between the accuracies of `LMIBPI` versus `RankTr`, `SVM+`, `SVM2k-LUPI` and `KCCA-LUPI`. These plots highlight that `LMIBPI` compares favorably. Figure 4.12 shows how `LMIBPI` compares against `LB-LMIBPI` and `UB-LMIBPI`. On average, using the auxiliary information allows recovering 41.6% of the performance gap in the linear kernel case, and 50.7% when HIK is used. Also, using HIK improves the average performance from $61.42 \pm 4.60\%$ to $65.50 \pm 3.55\%$.

Figure 4.13 gives a comprehensive outlook of how `LMIBPI` compares with the other single-view and LUPI classifiers. In general, `LMIBPI` outperforms the competition. For all the three datasets, HMDB, ImageNet, and CGD2011, using a non-linear kernel has led to a performance improvement, and for ImageNet and CGD2011 this has also led to a further performance improvement of `LMIBPI` with respect to the other approaches.

Finally, for scientific honesty, we note that in 2 cases for HMDB with HIK kernel, 2 cases for ImageNet with linear kernel, and 1 case for CGD2011 with linear kernel, on average, `LMIBPI` performs slightly below `LB-LMIBPI`. These are a few unfortunate cases where the optimization on average converges to a local minimum that leads to a worse solution than the corresponding single-view classifier.

AwA dataset: We use the Animals with Attributes (AwA) dataset [199], which contains images of animal categories, and repeat the same experiment performed in [10, 9]. We use the 10 test classes for which the attribute annotations are provided, for a total of 6180 images. The attributes capture 85 properties of the animals. We use the same set of features used in [10]. The main view is given by L_1 normalized 2000 dimensional SURF descriptors, and the attributes are the auxiliary view obtained from the DAP model [199]. We train 45 binary classifiers for each class pair combination. We use 50 and 200 samples per class for training and testing, respectively. The train/test split is repeated 20 times. For fair comparison with [10, 9] we use the linear kernel. Table 4.7 reports only the average precision (AP) results, where we have indicated in bold when LMIBPI has improved the AP, which happens 20 times out of 45, and 12 times the improvement is significant according to the z-test. Finally, Table 4.8 repeats the same experiment of Table 4.7 where the linear kernel has been replaced with a Gaussian kernel. The average AP is 88.38 for the linear kernel and also for the Gaussian kernel. Therefore, switching to a non-linear kernel has not improved the performance.

		SVM	RankTr	SVM+	LIR	LMIBPI
1	*Chimpanzee versus Giant panda*	88.88 ± 0.51	**89.33 ± 0.50**	88.07 ± 0.57	88.28 ± 0.47	88.32 ± 0.33
2	*Chimpanzee versus Leopard*	93.74 ± 0.26	93.70 ± 0.23	93.49 ± 0.29	93.36 ± 0.15	**94.05 ± 0.10**
3	*Chimpanzee versus Persian cat*	90.14 ± 0.40	91.00 ± 0.39	89.88 ± 0.42	**91.59 ± 0.40**	90.76 ± 0.19
4	*Chimpanzee versus Pig*	85.64 ± 0.57	86.08 ± 0.43	85.19 ± 0.53	83.74 ± 0.35	**87.32 ± 0.17**
5	*Chimpanzee versus Hippopotamus*	86.40 ± 0.55	86.92 ± 0.45	86.31 ± 0.59	89.63 ± 0.31	**90.21 ± 0.12**
6	*Chimpanzee versus Humpback whale*	98.03 ± 0.18	98.08 ± 0.18	97.74 ± 0.22	**98.30 ± 0.16**	97.76 ± 0.26
7	*Chimpanzee versus Raccoon*	87.01 ± 0.46	87.07 ± 0.48	86.64 ± 0.47	85.90 ± 0.63	**88.21 ± 0.27**
8	*Chimpanzee versus Rat*	85.42 ± 0.53	**86.67 ± 0.56**	84.83 ± 0.68	85.43 ± 0.48	85.31 ± 0.29
9	*Chimpanzee versus Seal*	91.74 ± 0.39	91.54 ± 0.43	91.10 ± 0.59	92.78 ± 0.42	**93.11 ± 0.23**
10	*Giant panda versus Leopard*	93.71 ± 0.38	93.76 ± 0.29	**94.03 ± 0.28**	92.81 ± 0.48	92.95 ± 0.20
11	*Giant panda versus Persian cat*	92.55 ± 0.41	92.57 ± 0.43	92.66 ± 0.32	**93.75 ± 0.29**	92.82 ± 0.32
12	*Giant panda versus Pig*	86.64 ± 0.45	86.22 ± 0.52	86.55 ± 0.40	84.19 ± 0.69	**86.71 ± 0.40**
13	*Giant panda versus Hippopotamus*	90.04 ± 0.56	90.89 ± 0.36	89.93 ± 0.56	**91.27 ± 0.35**	91.12 ± 0.29
14	*Giant panda versus Humpback whale*	98.38 ± 0.17	98.53 ± 0.15	98.11 ± 0.19	98.67 ± 0.11	**98.82 ± 0.14**
15	*Giant panda versus Raccoon*	**89.36 ± 0.44**	88.66 ± 0.60	89.06 ± 0.49	86.90 ± 0.74	89.21 ± 0.30
16	*Giant panda versus Rat*	88.49 ± 0.49	87.53 ± 0.51	87.86 ± 0.48	88.76 ± 0.37	**89.13 ± 0.25**
17	*Giant panda versus Seal*	92.81 ± 0.32	92.40 ± 0.40	92.59 ± 0.38	93.32 ± 0.31	**93.81 ± 0.19**
18	*Leopard versus Persian cat*	95.08 ± 0.25	**95.26 ± 0.25**	94.93 ± 0.24	**95.26 ± 0.22**	94.97 ± 0.22
19	*Leopard versus Pig*	88.55 ± 0.28	**88.90 ± 0.28**	88.37 ± 0.36	85.34 ± 0.50	87.31 ± 0.21
20	*Leopard versus Hippopotamus*	**92.98 ± 0.29**	92.86 ± 0.26	92.73 ± 0.31	92.54 ± 0.28	92.71 ± 0.16
21	*Leopard versus Humpback whale*	98.49 ± 0.30	98.63 ± 0.23	98.27 ± 0.31	**98.83 ± 0.11**	98.61 ± 0.26
22	*Leopard versus Raccoon*	80.31 ± 0.75	79.84 ± 0.59	79.94 ± 0.73	**81.31 ± 0.67**	80.12 ± 0.22
23	*Leopard versus Rat*	88.74 ± 0.35	89.27 ± 0.28	88.92 ± 0.35	89.93 ± 0.28	**90.13 ± 0.21**
24	*Leopard versus Seal*	93.87 ± 0.36	94.30 ± 0.36	93.74 ± 0.37	94.12 ± 0.21	**95.18 ± 0.33**
25	*Persian cat versus Pig*	81.55 ± 0.59	81.68 ± 0.46	81.45 ± 0.57	**82.60 ± 0.58**	82.27 ± 0.24
26	*Persian cat versus Hippopotamus*	92.42 ± 0.34	**92.82 ± 0.30**	92.33 ± 0.33	92.00 ± 0.49	92.38 ± 0.32
27	*Persian cat versus Humpback whale*	95.92 ± 0.29	95.84 ± 0.30	95.45 ± 0.38	97.36 ± 0.15	**97.42 ± 0.25**
28	*Persian cat versus Raccoon*	90.19 ± 0.40	90.38 ± 0.39	90.31 ± 0.41	**91.72 ± 0.34**	91.24 ± 0.18
29	*Persian cat versus Rat*	67.19 ± 0.60	69.07 ± 0.48	67.56 ± 0.63	69.62 ± 0.84	**70.49 ± 0.45**
30	*Persian cat versus Seal*	84.79 ± 0.60	85.66 ± 0.49	84.46 ± 0.54	88.38 ± 0.44	**88.41 ± 0.36**
31	*Pig versus Hippopotamus*	74.42 ± 0.48	75.57 ± 0.58	73.47 ± 0.55	**77.75 ± 0.51**	73.42 ± 0.12
32	*Pig versus Humpback whale*	96.01 ± 0.33	95.93 ± 0.37	95.75 ± 0.30	**96.85 ± 0.18**	95.93 ± 0.12
33	*Pig versus Raccoon*	77.73 ± 0.80	79.13 ± 0.63	76.96 ± 0.85	81.61 ± 0.71	**82.19 ± 0.15**
34	*Pig versus Rat*	68.66 ± 0.76	70.77 ± 0.73	68.58 ± 0.41	72.47 ± 0.55	**73.31 ± 0.25**
35	*Pig versus Seal*	77.91 ± 0.71	79.26 ± 0.77	77.32 ± 0.73	82.61 ± 0.55	**83.11 ± 0.43**
36	*Hippopotamus versus Humpback whale*	**92.19 ± 0.44**	92.17 ± 0.44	91.64 ± 0.60	91.08 ± 0.63	90.11 ± 0.28
37	*Hippopotamus versus Raccoon*	85.54 ± 0.60	**85.84 ± 0.70**	85.03 ± 0.60	85.72 ± 0.63	84.46 ± 0.36
38	*Hippopotamus versus Rat*	84.49 ± 0.39	85.62 ± 0.48	84.25 ± 0.37	85.91 ± 0.48	**86.11 ± 0.26**
39	*Hippopotamus versus Seal*	69.79 ± 0.83	**70.83 ± 0.79**	69.43 ± 0.84	69.79 ± 0.70	70.49 ± 0.41
40	*Humpback whale versus Raccoon*	96.67 ± 0.28	96.90 ± 0.29	96.57 ± 0.31	**97.34 ± 0.20**	96.97 ± 0.27
41	*Humpback whale versus Rat*	94.52 ± 0.19	**94.56 ± 0.22**	93.97 ± 0.24	92.95 ± 0.68	93.89 ± 0.19
42	*Humpback whale versus Seal*	84.60 ± 0.49	84.81 ± 0.38	84.24 ± 0.49	85.91 ± 0.57	**86.13 ± 0.17**
43	*Raccoon versus Rat*	77.65 ± 0.64	78.61 ± 0.72	78.36 ± 0.54	**80.00 ± 0.57**	79.63 ± 0.14
44	*Raccoon versus Seal*	91.43 ± 0.36	91.51 ± 0.40	91.37 ± 0.38	89.21 ± 0.43	**91.63 ± 0.36**
45	*Rat versus Seal*	78.45 ± 0.65	**79.88 ± 0.69**	78.28 ± 0.75	79.02 ± 0.50	79.21 ± 0.28
	Average	87.28	87.92	87.53	88.13	**88.38**

Table 4.7: AwA dataset. AP results for one-vs-one classification. Best average precisions are highlighted in boldface. The red boldface numbers indicate when the improvement of the LMIBPI method over the second best value was significant according to the z-test. The table refers to the case where we use 50 and 200 samples per class for training and testing, respectively. The values for the columns indicated as SVM, RankTr, SVM+, and LIR have been reported directly from [9] and from the supplementary material of [10].

This is likely due to the use of SURF features, which have a very high dimension, and are known to work well with linear kernels.

The table including the results of the other approaches can be found in [9]. Figure 4.7 shows that SVM has the highest AP 3 times, SVM+ 1 time, RankTr 9 times, and LIR [9] 12 times.

		SVM	SVM+	SVM2k-LUPI	KCCA-LUPI	LMIBPI
1	*Chimpanzee versus Giant panda*	**87.96 ± 0.18**	87.81 ± 0.08	87.89 ± 0.42	87.60 ± 0.20	86.48 ± 0.46
2	*Chimpanzee versus Leopard*	**95.15 ± 0.56**	92.12 ± 0.33	93.80 ± 0.29	92.39 ± 0.30	92.90 ± 0.26
3	*Chimpanzee versus Persian cat*	88.20 ± 0.13	92.30 ± 0.04	90.60 ± 0.57	90.83 ± 0.24	**93.42 ± 0.04**
4	*Chimpanzee versus Pig*	84.77 ± 0.25	85.57 ± 0.25	85.37 ± 0.15	85.09 ± 0.54	**86.29 ± 0.42**
5	*Chimpanzee versus Hippopotamus*	87.35 ± 0.06	86.83 ± 0.26	87.11 ± 0.29	87.72 ± 0.48	**89.84 ± 0.38**
6	*Chimpanzee versus Humpback whale*	98.25 ± 0.06	97.61 ± 0.10	**98.64 ± 0.39**	96.73 ± 0.31	98.19 ± 0.18
7	*Chimpanzee versus Raccoon*	83.60 ± 0.13	83.62 ± 0.16	86.07 ± 0.21	83.39 ± 0.49	**87.33 ± 0.42**
8	*Chimpanzee versus Rat*	85.52 ± 0.24	85.74 ± 0.54	86.50 ± 0.07	85.33 ± 0.26	**88.15 ± 0.42**
9	*Chimpanzee versus Seal*	90.78 ± 0.39	90.10 ± 0.30	89.96 ± 0.07	**90.82 ± 0.13**	88.62 ± 0.31
10	*Giant panda versus Leopard*	95.49 ± 0.49	**95.69 ± 0.14**	93.83 ± 0.13	91.79 ± 0.03	95.40 ± 0.41
11	*Giant panda versus Persian cat*	90.85 ± 0.00	**93.64 ± 0.30**	93.16 ± 0.46	90.63 ± 0.38	93.47 ± 0.29
12	*Giant panda versus Pig*	86.07 ± 0.52	86.09 ± 0.35	86.44 ± 0.54	**87.75 ± 0.15**	85.19 ± 0.58
13	*Giant panda versus Hippopotamus*	91.31 ± 0.29	90.85 ± 0.40	91.88 ± 0.50	92.24 ± 0.29	**92.83 ± 0.14**
14	*Giant panda versus Humpback whale*	**99.45 ± 0.49**	98.34 ± 0.39	98.74 ± 0.26	97.48 ± 0.30	98.20 ± 0.53
15	*Giant panda versus Raccoon*	88.18 ± 0.30	90.36 ± 0.55	**90.43 ± 0.09**	88.74 ± 0.16	89.54 ± 0.03
16	*Giant panda versus Rat*	84.63 ± 0.51	**88.18 ± 0.51**	87.41 ± 0.17	85.62 ± 0.30	86.93 ± 0.06
17	*Giant panda versus Seal*	90.07 ± 0.49	**91.30 ± 0.22**	90.86 ± 0.26	89.66 ± 0.30	90.09 ± 0.14
18	*Leopard versus Persian cat*	**95.09 ± 0.58**	93.38 ± 0.55	94.40 ± 0.06	94.82 ± 0.00	93.85 ± 0.07
19	*Leopard versus Pig*	**88.62 ± 0.12**	87.31 ± 0.10	88.10 ± 0.18	85.20 ± 0.31	86.81 ± 0.09
20	*Leopard versus Hippopotamus*	91.04 ± 0.04	91.51 ± 0.20	92.39 ± 0.53	90.74 ± 0.18	**94.06 ± 0.26**
21	*Leopard versus Humpback whale*	97.02 ± 0.00	98.21 ± 0.14	**99.04 ± 0.49**	98.19 ± 0.47	98.94 ± 0.05
22	*Leopard versus Raccoon*	**82.90 ± 0.11**	80.42 ± 0.29	81.46 ± 0.19	82.31 ± 0.13	82.65 ± 0.54
23	*Leopard versus Rat*	85.85 ± 0.52	87.91 ± 0.51	86.87 ± 0.19	87.65 ± 0.34	**88.85 ± 0.08**
24	*Leopard versus Seal*	93.02 ± 0.46	92.21 ± 0.12	94.13 ± 0.48	95.08 ± 0.48	**95.13 ± 0.04**
25	*Persian cat versus Pig*	**82.51 ± 0.27**	79.40 ± 0.05	80.76 ± 0.14	81.12 ± 0.01	81.94 ± 0.14
26	*Persian cat versus Hippopotamus*	92.02 ± 0.36	**94.00 ± 0.13**	92.83 ± 0.37	93.67 ± 0.26	93.56 ± 0.51
27	*Persian cat versus Humpback whale*	94.90 ± 0.23	98.37 ± 0.23	97.06 ± 0.08	**99.01 ± 1.00**	96.18 ± 0.18
28	*Persian cat versus Raccoon*	90.33 ± 0.23	**92.18 ± 0.39**	90.76 ± 0.51	90.70 ± 0.17	91.71 ± 0.39
29	*Persian cat versus Rat*	62.81 ± 0.21	63.04 ± 0.04	65.04 ± 0.37	64.07 ± 0.04	**67.80 ± 0.11**
30	*Persian cat versus Seal*	87.09 ± 0.44	87.47 ± 0.18	86.34 ± 0.49	87.29 ± 0.30	**89.21 ± 0.36**
31	*Pig versus Hippopotamus*	75.30 ± 0.06	76.38 ± 0.50	**76.71 ± 0.19**	74.16 ± 0.49	76.56 ± 0.48
32	*Pig versus Humpback whale*	94.30 ± 0.18	94.42 ± 0.30	96.74 ± 0.00	**97.13 ± 0.36**	96.28 ± 0.16
33	*Pig versus Raccoon*	79.55 ± 0.09	78.02 ± 0.15	80.07 ± 0.16	79.55 ± 0.05	**81.39 ± 0.49**
34	*Pig versus Rat*	70.72 ± 0.33	70.25 ± 0.41	72.98 ± 0.33	74.23 ± 0.19	**74.68 ± 0.11**
35	*Pig versus Seal*	77.88 ± 0.16	78.72 ± 0.20	78.08 ± 0.32	78.56 ± 0.56	**79.66 ± 0.07**
36	*Hippopotamus versus Humpback whale*	92.54 ± 0.17	93.70 ± 0.14	92.62 ± 0.00	94.02 ± 0.54	**94.15 ± 0.28**
37	*Hippopotamus versus Raccoon*	88.30 ± 0.17	83.36 ± 0.09	86.36 ± 0.47	87.68 ± 0.46	**89.60 ± 0.50**
38	*Hippopotamus versus Rat*	84.01 ± 0.59	**85.90 ± 0.17**	83.95 ± 0.05	83.59 ± 0.12	82.86 ± 0.06
39	*Hippopotamus versus Seal*	71.13 ± 0.49	**72.74 ± 0.03**	71.73 ± 0.47	69.87 ± 0.46	70.76 ± 0.50
40	*Humpback whale versus Raccoon*	96.57 ± 0.45	**98.59 ± 0.43**	97.08 ± 0.40	96.96 ± 0.28	96.29 ± 0.28
41	*Humpback whale versus Rat*	**96.54 ± 0.38**	96.01 ± 0.34	95.32 ± 0.29	95.25 ± 0.10	94.65 ± 0.04
42	*Humpback whale versus Seal*	84.21 ± 0.41	86.30 ± 0.38	85.97 ± 0.52	83.39 ± 0.31	**87.58 ± 0.10**
43	*Raccoon versus Rat*	78.09 ± 0.04	78.27 ± 0.28	77.96 ± 0.43	78.61 ± 0.02	**79.33 ± 0.38**
44	*Raccoon versus Seal*	92.05 ± 0.42	**92.51 ± 0.34**	90.60 ± 0.20	91.44 ± 0.52	89.36 ± 0.08
45	*Rat versus Seal*	77.29 ± 0.19	79.03 ± 0.38	77.39 ± 0.19	77.29 ± 0.02	**80.54 ± 0.44**
	Average	87.32	87.75	87.81	87.45	**88.38**

Table 4.8: AwA dataset. AP results for one-vs-one classification. Best average precisions are highlighted in boldface. The values in this table have been obtained by conducting the same experiment performed to obtain Table 4.7, with the difference that a Gaussian kernel was used in place of the linear kernel.

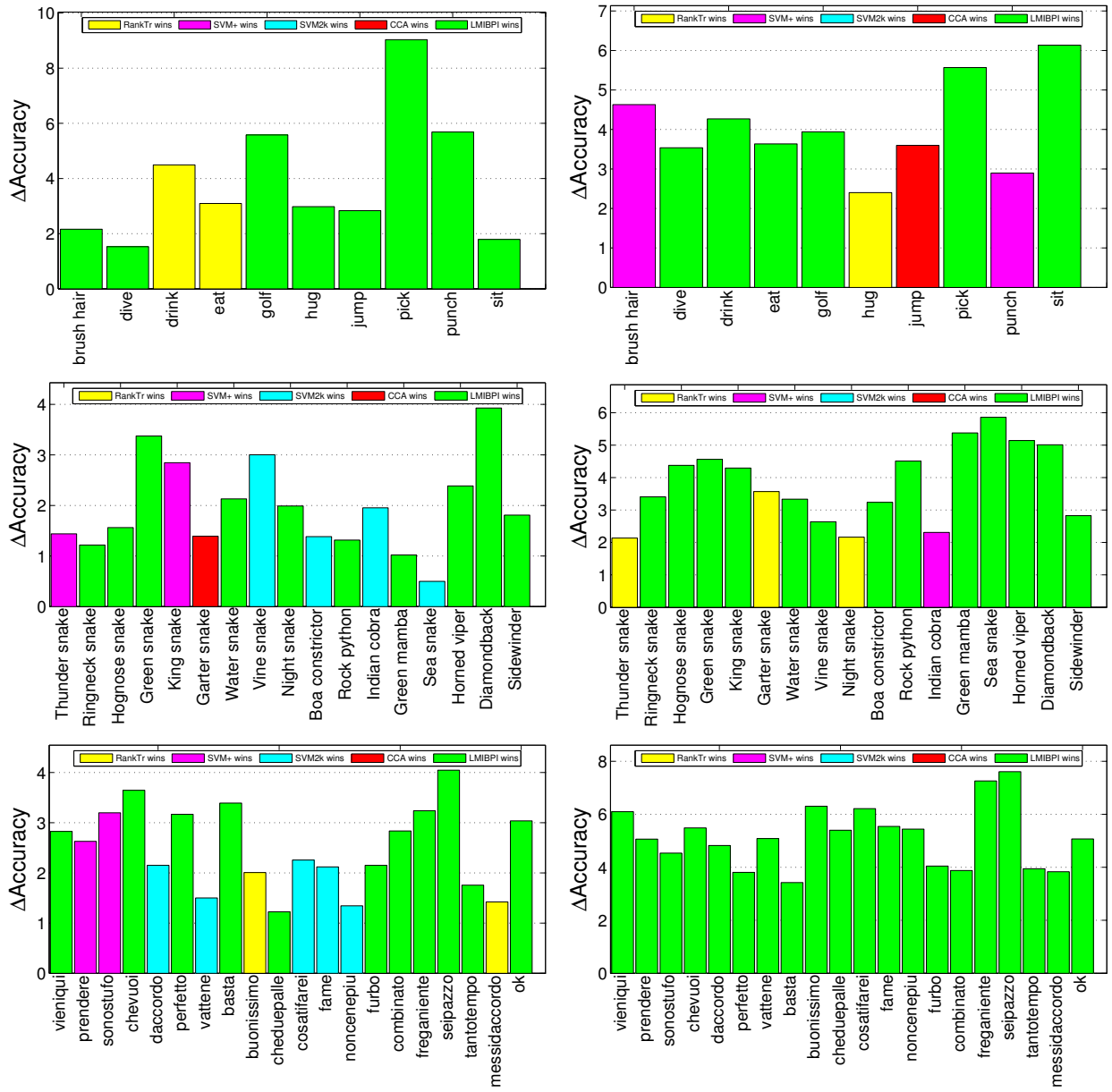

Figure 4.4: Linear vs. non-linear kernel. Plots representing the differences between the classification accuracy of the winner LUPI method against the average accuracy over the following methods: `RankTr` (yellow), `SVM+` (magenta), `SVM2k-LUPI` (cyan), `KCCA-LUPI` (red), and `LMIBPI` (green). The linear kernel was used on the left plots, and the histogram intersection kernel on the right plots. The top row come from the HMDB dataset, the middle row from ImageNet, and the last row from CGD2011.

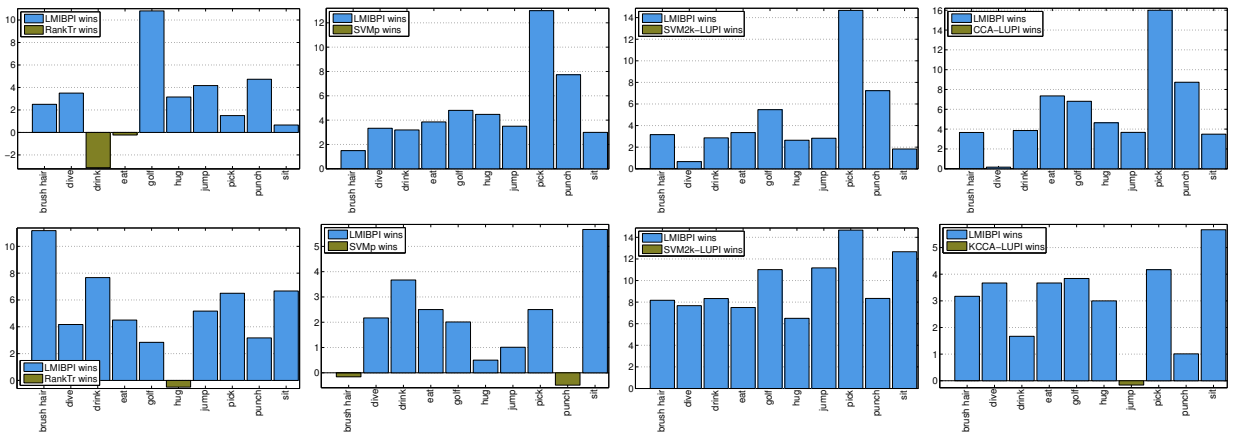

Figure 4.5: HMDB dataset. Each row shows the plots of the differences between the classification accuracy of `LMIBPI` versus `RankTr`, `SVM+`, `SVM2k-LUPI`, and `CCA-LUPI`, respectively. The top row refers to the use of the linear kernel. The bottom row refers to the use of the HIK kernel.

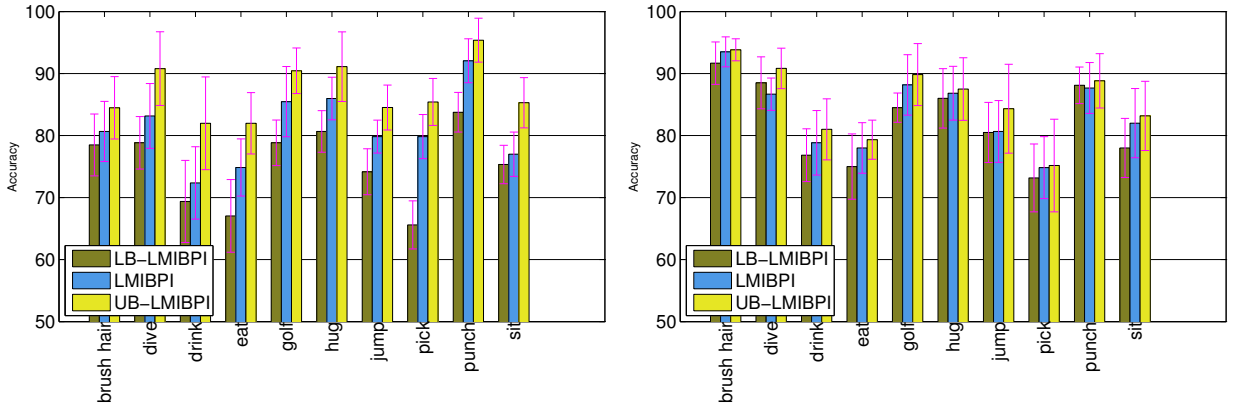

Figure 4.6: HMDB dataset. Comparison between the classification accuracy of `LMIBPI` versus the corresponding single-view classifier (lower bound) `LB-LMIBPI`, and two-view classifier (upper bound) `UB-LMIBPI`. The left plot refers to the use of the linear kernel. The right plot refers to the use of the HIK kernel.

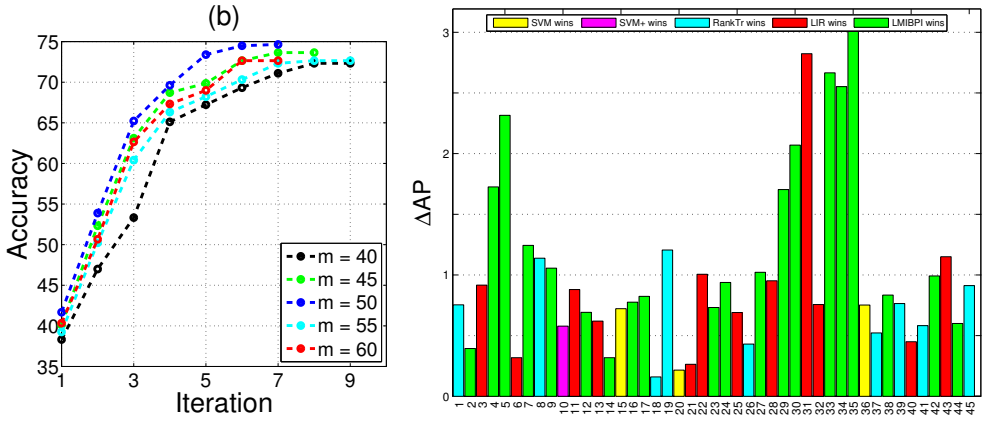

Figure 4.7: Rate of convergence and AwA dataset. Left: Plot showing the convergence rate for different m's for the `drink` class of the HMDB dataset. Right: Plot showing the differences between the AP of the winner LUPI method against the average accuracy over the following methods: `SVM` (yellow), `SVM+` (magenta), `RankTr` (cyan), `LIR` (red), and `LMIBPI` (green).

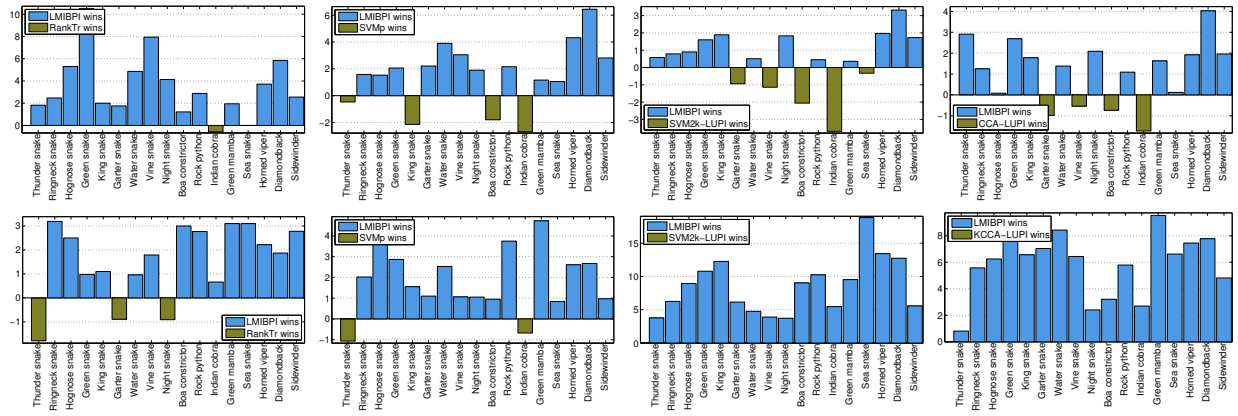

Figure 4.8: ImageNet dataset. Each row shows the plots of the differences between the classification accuracy of `LMIBPI` versus `RankTr`, `SVM+`, `SVM2k-LUPI`, and `CCA-LUPI`, respectively. The top row refers to the use of the linear kernel. The bottom row refers to the use of the HIK kernel.

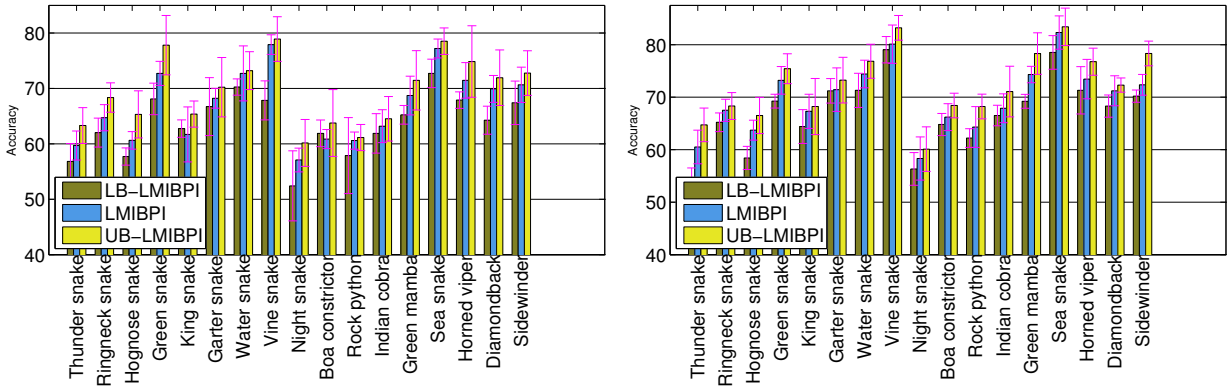

Figure 4.9: ImageNet dataset. Comparison between the classification accuracy of `LMIBPI` versus the corresponding single-view classifier (lower bound) `LB-LMIBPI`, and two-view classifier (upper bound) `UB-LMIBPI`. The left plot refers to the use of the linear kernel. The right plot refers to the use of the HIK kernel.

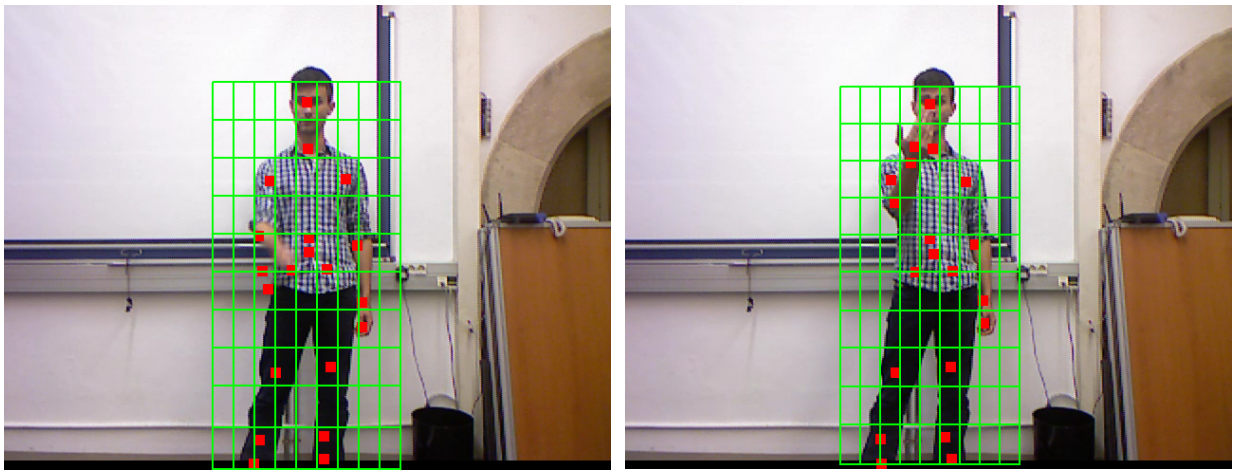

Figure 4.10: CGD2011 dataset. Samples from the CGD2011 dataset with joint information superimposed (red squares), together with a (green) grid visualizing the binning of the joint positions.

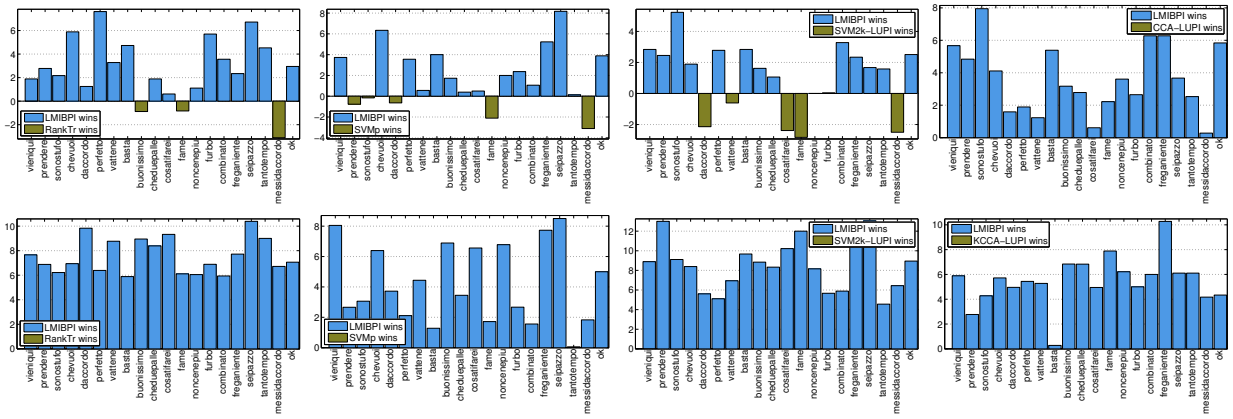

Figure 4.11: CGD2011 dataset. Each row shows the plots of the differences between the classification accuracy of `LMIBPI` versus `RankTr`, `SVM+`, `SVM2k-LUPI`, and `CCA-LUPI`, respectively. The top row refers to the use of the linear kernel. The bottom row refers to the use of the HIK kernel.

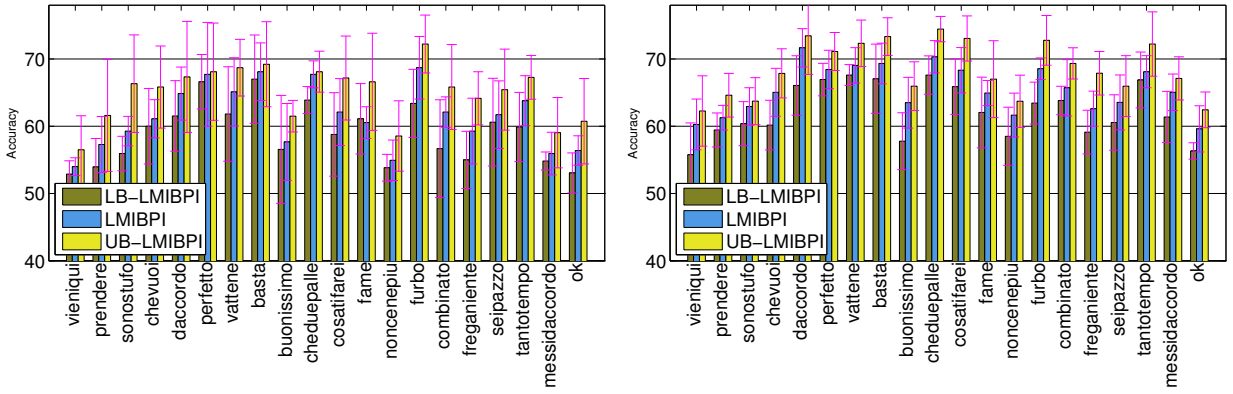

Figure 4.12: CGD2011 dataset. Comparison between the classification accuracy of `LMIBPI` versus the corresponding single-view classifier (lower bound) `LB-LMIBPI`, and two-view classifier (upper bound) `UB-LMIBPI`. The left plot refers to the use of the linear kernel. The right plot refers to the use of the HIK kernel.

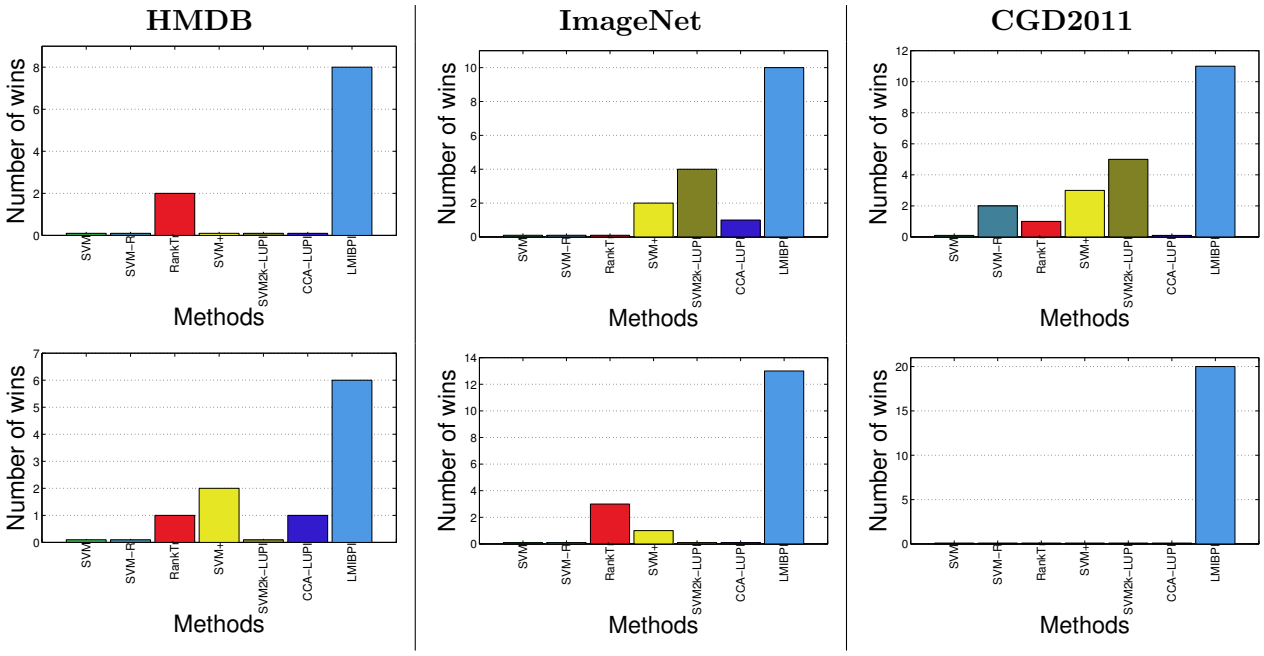

Figure 4.13: HMDB, ImageNet and CGD2011 datasets. Plots representing the number of wins accumulated by the single-view and the LUPI classifiers, namely `SVM`, `SVM-R`, and `RankTr`, `SVM+`, `SVM2k-LUPI`, `KCCA-LUPI`, `LMIBPI`. The top row refers to the use of the linear kernel. The bottom row refers to the use of the HIK kernel. The first column refers to the HMDB dataset. The middle column refers to the ImageNet dataset. The right column refers tot he CGD2011 dataset.

Chapter 5

Information Bottleneck Domain Adaptation with Privileged Information

We addressed the auxiliary view problem in Section 4. In this section, we take another step forward and consider the the auxiliary view problem and the unsupervised domain adaptation (UDA) problem jointly by taking an information theoretic approach. [80]. This is important because target distribution most likely is different than source distributions.

We are given a training dataset made of triplets $(x_1, x_1^*, y_1), \cdots, (x_N, x_N^*, y_N)$. The feature $x_i \in \mathcal{X}$ is a realization from a random variable X, the feature $x_i^* \in \mathcal{X}^*$ is a realization from a random variable X^*, and the label $y_i \in \mathcal{Y}$ is a realization from a random variable Y. The triplets are i.i.d. samples from a joint probability distribution $p(X, X^*, Y)$. In addition, we are given the data $x_1^t, \cdots, x_M^t$, where $x_i^t \in \mathcal{X}$ is a realization from a random variable X^t, and the data points are i.i.d. samples from a distribution $p(X^t)$. We assume that there is a *covariate shift* [52] between X and X^t, i.e., there is a difference between $p(X)$ and $p(X^t)$. We say that X represents the *main data view*, that X^* represents the *auxiliary data view*, and that X^t represents the *target data view*. The main and auxiliary views represent the *source domain*, and the target view the *target domain*. Under this settings the goal is to learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$ that during testing is going to perform well on data from the target view.

The problem just described (See Figure 5.1 for model overview) is different from the traditional

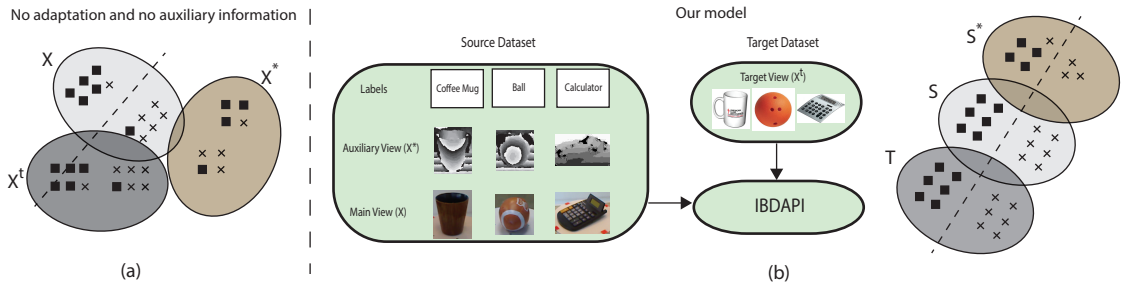

Figure 5.1: Domain Adaptation with Privileged Information We assumed that X represents the source/main samples, X^t, and X^* represent the extra information available in training from target and source samples, respectively. (a) Since target data distribution $p(X^t)$, and source data distribution $p(X)$ differ by a *covariate shift*, the classifier boundary is suboptimal. Even more so because the paired source auxiliary/privileged data X^* is not used for training. (b) Labeled paired source auxiliary/privileged data (e.g., depth data) is used, along with unlabeled target data, to improve visual recognition on the target domain via the *information bottleneck domain adaptation with privileged information (IBDAPI)* principle (as we will discuss in Chapter 5). IBDAPI learns a compressed representation where the mapped source data (S and S^*), as well as the mapped target data (T) become more separable.

unsupervised domain adaptation (UDA), because we also aim at exploiting the auxiliary data view during training for learning a better prediction function. On the other hand, the presence of the auxiliary view is reminiscent of the Learning Using Privileged Information (LUPI) paradigm as defined in [56], but there is a fundamental difference. In the LUPI framework the prediction function is used only on the main view, and the domain adaptation task is absent. While it has been shown that auxiliary data improves the performance of a traditional classifier [136], how to best carry this improvement over to a new target domain is still an open problem.

5.1 IB for UDA with auxiliary data

We use the IB framework of Section 4.2 to develop a new information bottleneck principle, which simultaneously accounts for the use of auxiliary data, as well as the adaptation to a new target domain. Specifically, let us assume that X, X^*, X^t and Y are four random variables with known distribution $p(X, X^*, X^t, Y)$. We develop the principle in two steps. First, we assume that the target view is an additional view of the source domain, and we extend the IB method to handle the auxiliary the main and the target views in the source, and the main and target views in the target domain. Then, we assume that the target view does not carry information about Y, and we address the covariate shift.

5.1.1 Incorporating auxiliary data

We assume that both X, X^*, and X^t carry information about Y. In addition, only the information carried by X and X^t can be used to predict Y. We want to design a principle for learning a model for prediction that also exploits the information carried by X^*.

The straightforward application of the IB method suggests to compress X into a latent variable S, and X^t into a latent variable T, as much as possible, while making sure that information about Y is retained. These two competing goals are depicted by the graphs G_{in} and G_{out} in Figures 5.2(a) and 5.2(b). Therefore, the IB method would seek for the optimal representation given by $q(X^t, X, X^*, Y, S, T) = q(S, T|X, X^t)p(X^t, X, X^*, Y)$, where $q(S, T|X, X^t)$ is such that $I(X; Y|S)$ and $I(X^t; Y|T)$ are as close to zero as possible. On the other hand, since X^* has knowledge about Y (as highlighted by the connection $X^* \to Y$ in G_{in}), we observe that $I(X^*; Y|S)$ and $I(X^*; Y|T)$ could be arbitrarily high. This means that knowing S and T still leaves with X^* substantial information about Y.

We address the problem just outlined by modifying G_{out} as in Figure 5.2(c), where the edges $S \to X^*$ and $T \to X^*$ have been added. In this way, knowing S and T makes not only X and Y independent, as well as X^t and Y, but also makes X^* and Y independent. This also means that the optimal $q(S, T|X, X^t)$ will minimize $I(X; Y|S)$ and $I(X^t; Y|T)$, as well as $I(X^*; Y|S)$ and $I(X^*; Y|T)$. In particular, the multi-informations of G_{in} and G_{out} in Figures 5.2(a) and 5.2(c) are given by

$$\mathcal{I}^{G_{in}} = I(S; X) + I(T; X^t) + I(Y; X^t, X, X^*) \,, \tag{5.1}$$

$$\mathcal{I}^{G_{out}} = I(S; X) + I(T; X^t) + I(S, T; X^*) + I(S, T; Y) \,. \tag{5.2}$$

By plugging (5.1) and (5.2) into (4.2), since $I(Y; X^t, X, X^*)$ is constant, the functional for learning the optimal representation for S and T is given by

$$\mathcal{L}[q(S, T|X, X^t)] = I(S; X) + I(T; X^t) - \gamma I(S, T; X^*) - \gamma I(S, T; Y) \,, \tag{5.3}$$

where γ strikes a balance between compressing X and X^t and imposing the independency requirements.

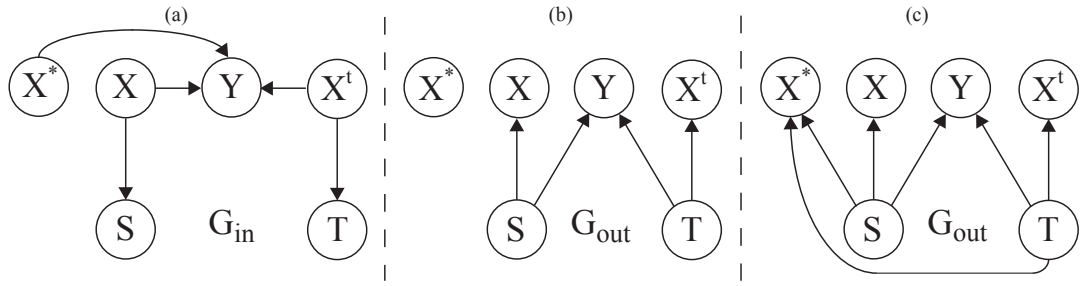

Figure 5.2: Information Bottleneck with Auxiliary Data. Structural representation of G_{in} (a), and G_{out} (b,c) used by the information bottleneck method. (b) G_{out} does not leverage the auxiliary data. (c) G_{out} leverages the auxiliary data.

5.1.2 Adapting to a new target domain

Model (5.3) incorporates the target view X^t under the assumption that it can predict the relevant information Y. This implies a fully supervised scenario, where training data should be given in quadruplets, i.e., (x_i^t, x_i, x_i^*, y_i). On the other hand, we are interested in the unsupervised setting, where the training target view is not labeled and not paired with the source data. From a statistical point of view, this assumption corresponds to saying that $p(X^t, X, X^*, Y) = p(X^t)p(X, X^*, Y)$, which leads to a number of consequences. First, the graph G_{in} of Figure 5.2(a) now becomes as in Figure 5.3(a), where we do not consider the dotted edges for the moment. In addition, it is easy to show that $I(S, T; X^*) = I(S; X^*)$, and that $I(S, T; Y) = I(S; Y)$. Therefore, the graph structure G_{out} in Figure 5.2(c) now becomes as in Figure 5.3(b). Finally, it is also easy to show that $q(S, T|X, X^t) = q(S|X)q(T|X^t)$. Therefore, the *unsupervised* scenario reduces model (5.3) to the following

$$\mathcal{L}[q(S|X), q(T|X^t)] = I(S; X) + I(T; X^t) - \gamma I(S; X^*) - \gamma I(S; Y) \ . \tag{5.4}$$

We note that estimating the optimal compressed representation S and T of X and X^t, by minimizing (5.4) leads to an ill-posed problem. This is because at convergence $q(T|X^t)$ would simply minimize $I(T; X^t)$. On the other hand, we are interested in addressing the distribution mismatch between the main view and the target view. Therefore, rather than treating $q(S|X)$ and $q(T|X^t)$ as separate free functions, we make the assumption that the compression maps from the main and the target views should cause $q(S|X)$ and $q(T|X^t)$ to be the same, in order to minimize the covariate shift in the compressed domain. If we restrict the search for the optimal representation

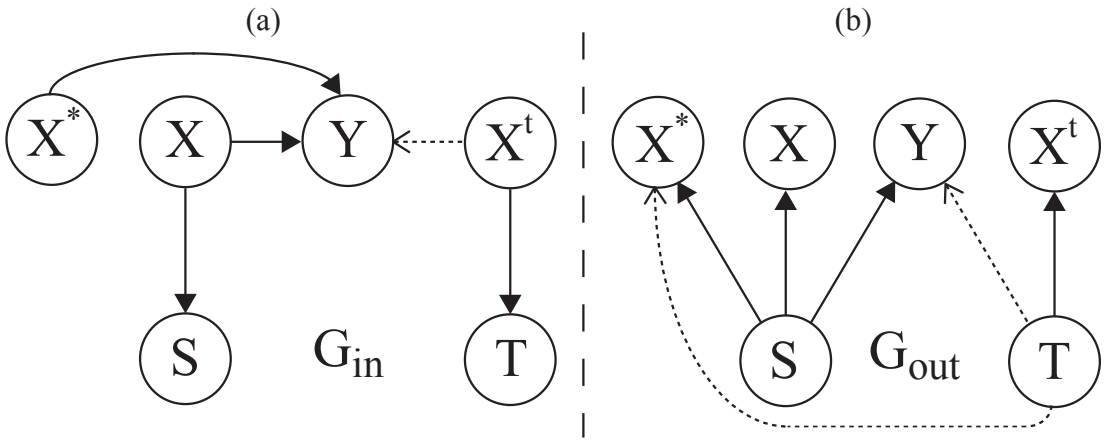

Figure 5.3: Information Bottleneck Domain Adaptation with Privileged Information. Structural representation of G_{in} and G_{out} used by the IBDAPI principle (5.5)**.**

within a family of distributions parameterized by A, this means that $q(S|X) \doteq q_A(S|X)$ and $q(T|X^t) \doteq q_A(T|X^t)$, i.e., they should have the same parameter. This assumption would impose $q(S|X)$ and $q(T|X^t)$ to no longer be independent, and therefore all the consequences originated by the statistical independence of X^t would be reversed, to a certain extent. In other words, it would be as if the links $X^t \to Y$ in G_{in}, and $T \to X^*$ and $T \to Y$ in G_{out}, were partially restored, which is why they appear with dotted lines in Figure 5.3. Finally, this assumption reduces (5.4) to the proposed principle

$$\boxed{\mathcal{L}[q_A(\cdot|\cdot)] = I(S;X) + I(T;X^t) - \gamma I(S;X^*) - \gamma I(S;Y)} \tag{5.5}$$

Since the auxiliary view plays the role of privileged information, we call learning representations by minimizing the functional (5.5) as the *information bottleneck domain adaptation with privileged information (IBDAPI)*.

5.2 IBDAPI for visual recognition

Our goal is to design a framework for visual recognition, where a classification task is based on the *target* view X^t of the visual data, for which some unlabeled samples are given for training. Moreover, at training time labeled samples from a *main* view X are also given, as well as some samples from an *auxiliary* view X^*. We pose no restrictions on the type of auxiliary data available.

The IBDAPI method (5.5) learns how to compress X and X^t into S and T in a way that is optimal for predicting Y (representing class labels), but also that best exploits the information carried by X^* about Y. Therefore, T appears to be the representation of choice for predicting Y. However, while IBDAPI provides for a compression map defined explicitly by $q_A(\cdot|\cdot)$, the prediction map for doing classification, identified by $q(Y|S)$ is much harder to compute in general. This is why we modify the IBDAPI method into one that is tailored to visual recognition.

We note that the last term in (5.5) is equivalent to the constraint $I(S;Y) \geq \textit{constant}$ if γ is interpreted as a Lagrange multiplier. This means that S should carry at least a certain amount of information about Y. On the other hand, we are interested in learning a decision function $f : \mathcal{S} \rightarrow \mathcal{Y}$ that uses such information for classification purposes. Therefore, we replace the constraint on $I(S;Y)$ with the risk associated to $f(S)$ according to a loss function ℓ. Thus, for visual recognition, (5.5) is modified into

$$\boxed{\mathcal{L}[q_A(\cdot|\cdot), f] = I(S;X) + I(T;X^t) - \gamma I(S;X^*) + \beta E[\ell(f(S), Y)]} \tag{5.6}$$

where $E[\cdot]$ denotes statistical expectation, and β balances the risk versus the compression requirements. Note that the modified IBDAPI criterion (5.6) is general, and could be used with any classifier.

5.2.1 Large-margin IBDAPI

We use (5.6) for learning a multi-class large-margin classifier. We parameterize the search space for $q_A(\cdot|\cdot)$ by assuming $S = \phi(X;A)$, as well as $T = \phi(X^t;A)$, where A is a suitable set of parameters. Moreover, $f(S)$ is a k-class decision function given by $Y = \arg\max_{m=1,\cdots,k} \langle w_m, S\rangle$, where $\langle\cdot,\cdot\rangle$ identifies a dot product, and $W = [w_1, \cdots, w_k]$ defines a set of margins. Therefore, based on [200], (5.6) leads to the following classifier learning formulation, which we refer to as the *large-margin IBDAPI (LMIBDAPI)*

$$\begin{aligned} &\min_{A,W,\xi_i} I(S;X) + I(T;X^t) - \gamma I(S;X^*) + \frac{\beta}{2}\|W\|_2^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \\ &\text{s.t.} \quad \langle w_{y_i} - w_m, \phi(x_i, A)\rangle \geq e_i^m - \xi_i \ , \ \ \xi_i \geq 0 \ , \ \ m = 1, \cdots, k \ , \ \ i = 1, \cdots, N \ . \end{aligned} \tag{5.7}$$

where $e_i^m = 0$ if $y_i = m$ and $e_i^m = 1$ otherwise. ξ_i indicates the usual slack variables, and C is the usual parameter to control the slackness.

Kernels. As described in Section 4.4.1, we set $S = \phi(X, A) = A\phi(X)$, and $T = \phi(X^t, A) = A\phi(X^t)$, where we require $\phi(X)$ and $\phi(X^t)$ to have positive components and be normalized to 1, and A to be a stochastic matrix, made of conditional probabilities between components of $\phi(X)$ ($\phi(X^t)$) and S (T). This assumption greatly simplifies computing mutual informations. This mapping also allows the use of kernels. X^* is mapped to a feature space with the same requirements by using the same strategy. Thus, without loss of generality, in the sequel we set $S = AX$, and $T = AX^t$.

Mutual informations. $I(S; X)$ and $I(T; X^t)$ are given by

$$I(S;X) = E\left[\sum_{i,j} A(i,j)X(j)\log\frac{A(i,j)}{S(i)}\right] \quad I(T;X^t) = E\left[\sum_{i,j} A(i,j)X^t(j)\log\frac{A(i,j)}{T(i)}\right] \tag{5.8}$$

where $A(i,j)$ is the entry of A in position i,j, whereas $S(i)$ and $X(j)$ ($T(i)$ and $X^t(j)$) are the components in position i and j of S and X (T and X^t) respectively. Obviously, during training the expectation is replaced by the empirical average. To compute $I(S; X^*)$, it is easy to show that

$$I(S;X^*) = E\left[\sum_{i,j} A(i,\cdot)F(\cdot,j)X^*(j)\log\frac{A(i,\cdot)F(\cdot,j)}{S(i)}\right] \tag{5.9}$$

where F is also a stochastic matrix such that $X = FX^*$. F can be learned from the source training data with a projected gradient method [187], as described in Section 4.4.1.

Missing auxiliary views. Training samples with missing auxiliary view affect only $I(S; X^*)$. The issue is seamlessly handled by estimating F and the average in (5.9) by using only the samples that have the auxiliary view.

Optimization. When A is known, (5.7) is a soft-margin SVM problem. Instead, when the SVM parameters are known, (5.7) becomes

$$\begin{aligned} \min_A \quad & I(S;X) + I(T;X^t) - \gamma I(X^*;S) + \frac{C}{N}\sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & \xi_i = \max_{m=1,\cdots,k} \{\langle w_m - w_{y_i}, \phi(x_i, A)\rangle + e_i^m\} \ . \end{aligned} \tag{5.10}$$

Since the soft-margin problem is convex, if also (5.10) is convex, then an alternating direction method is guaranteed to converge. In general, the mutual informations in (5.10) are convex functions of $q(S|X)$ and $q(T|X^t)$ [186], while within a range of γ's the third mutual information leaves the sum of the three to be convex. The last term is also convex, however, the constraints define a non-convex set due to the discontinuity of the hinge loss function. Smoothing the hinge loss turns (5.10) into a convex problem, and allows to use an alternating direction method with variable splitting combined with the augmented Lagrangian method. This is done by setting $f(A) = I(S;X) + I(T;X^t) - \gamma I(X^*;S)$, $g(B) = \frac{C}{N}\sum_{i=1}^{N}\xi_i$, and then solving $\min_A\{f(A) + g(B) : A - B = 0\}$.

For smoothing the hinge loss we use the Nesterov smoothing technique [189]. Since the objective is to smooth $g(B)$, we proceed by relaxing its minimization into the sum of the minima of the slack variables. Doing so gives $\bar{g}(B)$, the smoothed version of $g(B)$, expressed as

$$\bar{g}(B) = \frac{C}{N}\sum_{i=1}^{N}\mu\ln(\frac{1}{m}\sum_{m=1}^{k}\cosh(\frac{1}{\mu}(\langle w_m - w_{y_i}, \phi(x_i, B)\rangle - e_i^m))) \tag{5.11}$$

and μ is a smoothing parameter. In this way, the minimization can be carried out with the Fast Alternating Linearization Method (FALM) [191]. This allows simpler computations, and has performance guarantees when ∇f and $\nabla\bar{g}$ are Lipschitz continuous, which is the case, given the smoothing technique that we have used. Since the procedure is almost identical to the Section 4, we refer the readers to that section.

In summary, we provide an optimization procedure guaranteed to converge, which starts by learning F. Then, until convergence alternates between learning a SVM, and solving (5.10). Note that this iterative optimization is fully conducted in the primal space for best computational efficiency.

5.3 Experiments

We have performed experiments on several datasets for object and gender recognition, and have compared our approach with several others summarized as follows.

Single-view classifiers: Using only the main view, we use libSVM [201] and LIBLINEAR [202] (indicated as `SVM`) for training binary and multi-class SVM classifiers.

Table 5.1: RGB-D-Caltech256 dataset. Classification accuracies for one-vs-all binary classifications with linear kernels. Main and auxiliary views are KDES features of the RGB and depth of the RGB-D Object dataset [11]. KDES features from the Caltech256 dataset [12] represent the target domain.

		MV and LUPI				UDA				UDA+LUPI	
	SVM	SVM2k	KCCA	SVM+	RankTr	SGF	LMK	SA	LMIBDA	DA-M2S	LMIBDAPI
Calculator	49.83 ± 1.65	50.08 ± 1.87	48.10 ± 2.58	54.61 ± 3.37	53.27 ± 1.26	54.23 ± 1.26	53.71 ± 2.78	54.22 ± 3.32	56.33 ± 2.78	55.63 ± 2.89	**59.52 ± 2.18**
Cereal box	69.10 ± 3.41	67.10 ± 3.60	67.40 ± 3.20	62.78 ± 3.53	63.26 ± 4.98	65.23 ± 3.25	66.81 ± 2.59	67.17 ± 3.89	67.92 ± 2.11	68.50 ± 4.27	**72.60 ± 2.63**
Coffee mug	57.95 ± 3.03	57.61 ± 3.97	57.13 ± 5.99	58.32 ± 3.45	58.36 ± 3.69	66.23 ± 4.21	67.36 ± 3.89	68.12 ± 5.11	68.36 ± 3.11	70.11 ± 5.19	**75.65 ± 3.39**
Keyboard	60.79 ± 6.04	59.77 ± 6.41	59.40 ± 6.08	58.21 ± 3.88	57.98 ± 3.48	61.59 ± 3.27	59.26 ± 3.89	62.65 ± 3.14	63.36 ± 3.25	63.52 ± 4.68	**68.50 ± 3.71**
Flashlight	72.06 ± 2.60	70.86 ± 3.95	70.56 ± 3.20	71.36 ± 2.21	70.68 ± 4.24	72.36 ± 2.78	70.26 ± 2.15	73.25 ± 2.68	72.15 ± 2.14	71.37 ± 2.78	**74.79 ± 2.51**
Lightbulb	67.09 ± 2.32	65.23 ± 2.71	66.69 ± 3.06	68.36 ± 3.77	67.58 ± 2.15	67.99 ± 1.89	66.36 ± 2.11	68.11 ± 1.67	67.23 ± 2.85	68.48 ± 3.81	**71.81 ± 1.49**
Mushroom	49.02 ± 4.45	51.41 ± 3.97	49.04 ± 3.54	54.71 ± 5.86	56.84 ± 4.15	66.36 ± 3.87	64.26 ± 4.15	68.22 ± 3.89	69.26 ± 3.14	70.00 ± 5.10	**70.39 ± 2.96**
Ball	45.19 ± 2.11	48.96 ± 0.78	45.05 ± 4.44	53.27 ± 1.84	54.48 ± 3.25	60.25 ± 2.11	61.36 ± 2.87	63.86 ± 1.89	64.95 ± 2.67	**67.27 ± 5.32**	65.45 ± 3.71
Soda can	52.04 ± 3.46	50.00 ± 3.30	50.09 ± 3.33	52.48 ± 3.76	50.26 ± 1.36	56.58 ± 2.18	55.71 ± 2.65	58.36 ± 2.14	60.33 ± 2.35	59.65 ± 2.63	**62.93 ± 2.84**
Tomato	56.05 ± 3.73	50.76 ± 0.99	53.69 ± 3.03	51.55 ± 3.71	50.23 ± 2.59	63.25 ± 2.17	64.25 ± 1.36	64.33 ± 2.74	64.26 ± 2.36	64.61 ± 3.19	**73.40 ± 2.22**
Average	57.91	57.18	56.71	58.56	58.29	63.41	62.93	64.83	65.42	65.91	**69.50**

		MV and LUPI			UDA			UDA+LUPI	
	SVM	SVM2k	KCCA	SVM+	LMK	SA	LMIBDA	DA-M2S	LMIBDAPI
Calculator	50.65 ± 2.65	51.68 ± 2.65	52.68 ± 3.69	56.69 ± 2.65	54.65 ± 2.69	56.26 ± 2.16	58.14 ± 1.59	57.69 ± 3.25	**61.36 ± 3.17**
Cereal box	68.69 ± 4.65	68.15 ± 2.16	66.39 ± 2.98	63.69 ± 2.15	65.65 ± 3.57	67.68 ± 2.59	68.06 ± 2.21	70.69 ± 3.25	**73.89 ± 2.15**
Coffee mug	58.96 ± 2.98	58.68 ± 2.69	56.25 ± 2.15	59.69 ± 3.69	59.15 ± 2.45	71.69 ± 3.69	70.15 ± 2.14	73.69 ± 4.65	**76.69 ± 3.67**
Keyboard	59.68 ± 5.24	58.12 ± 4.65	57.64 ± 5.17	61.65 ± 3.24	58.69 ± 3.84	63.69 ± 2.15	62.15 ± 2.58	64.15 ± 3.25	**67.65 ± 2.59**
Flashlight	67.69 ± 3.69	69.14 ± 2.69	69.56 ± 3.36	72.69 ± 2.15	72.64 ± 4.16	71.69 ± 1.71	73.15 ± 3.15	72.69 ± 1.25	**76.69 ± 3.15**
Lightbulb	67.69 ± 2.69	63.69 ± 3.69	67.68 ± 2.58	67.69 ± 3.15	68.69 ± 1.69	65.69 ± 2.26	66.15 ± 2.45	65.69 ± 2.69	**72.36 ± 1.20**
Mushroom	52.69 ± 4.98	53.69 ± 3.56	52.66 ± 2.68	56.36 ± 4.65	57.15 ± 3.45	67.36 ± 2.69	66.69 ± 2.78	**71.69 ± 4.69**	69.45 ± 2.69
Ball	48.69 ± 3.69	47.25 ± 2.68	46.54 ± 3.47	54.65 ± 2.15	53.24 ± 2.15	62.69 ± 3.45	63.78 ± 3.69	**69.69 ± 4.57**	68.69 ± 2.45
Soda can	54.98 ± 2.58	51.69 ± 4.64	51.45 ± 3.15	53.12 ± 2.15	51.68 ± 2.91	57.36 ± 3.71	58.18 ± 3.56	58.15 ± 2.98	**60.69 ± 2.45**
Tomato	55.69 ± 2.15	52.63 ± 2.65	54.65 ± 3.87	52.58 ± 2.11	51.45 ± 2.14	62.69 ± 2.48	65.97 ± 3.48	67.69 ± 2.78	**75.15 ± 3.11**
Average	58.54	57.47	57.55	59.88	59.29	64.68	65.24	67.18	**70.26**

Table 5.2: RGB-D-Caltech256 dataset. Classification accuracies for one-vs-all binary classifications with Gaussian kernels. Main and auxiliary views are KDES features of the RGB and depth of the RGB-D Object dataset [11]. KDES features from the Caltech256 dataset [12] represent the target domain.

LUPI and multi-view (MV) classifiers: By using the main and auxiliary views, we train the SVM+ [56] (indicated as `SVM+`, the Rank Transfer [10] (indicated as `RankTr`). We also train the SVM2k [73] and test only the SVM that uses the main view (indicated as `SVM2k`), and we perform kernel CCA (KCCA) [195] between main and auxiliary views, map the main view in feature space and train an SVM (indicated as `KCCA`). `SVM+`, `RankTr`, `SVM2k`, and `KCCA`, can be used only for binary classification.

UDA classifiers: We use the main view and the target training data for learning the Sampling Geodesic Flow (`SGF`) [97], the Landmark (`LMK`) [203], the Subspace Alignment (`SA`) [98], the Transfer Component Analysis (`TCA`) [114], and the Domain Invariant Projection (`DIP`) [83] classifiers. In addition, we use LMIBDAPI where we eliminate the auxiliary information by setting $\gamma = 0$ (indicated as `LMIBDA`).

UDA+LUPI classifiers: Besides our approach, indicated as `LMIBDAPI`, we consider the only other approach designed to work in the same settings, which is [59] (indicated as `DA-M2S`).

Model selection: We use the same joint cross validation and model selection procedure described

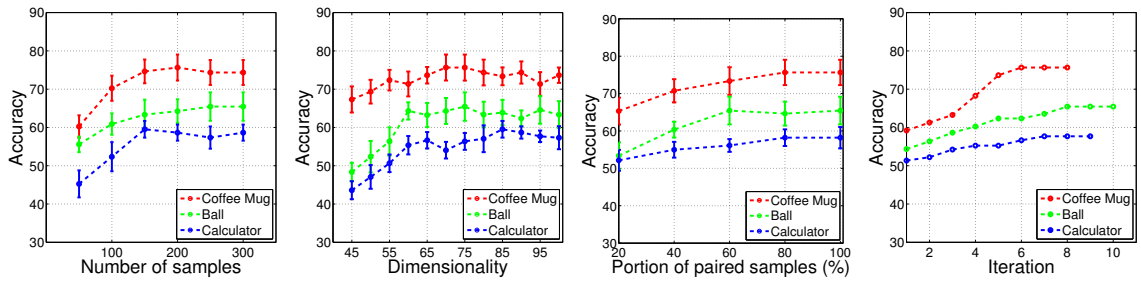

Figure 5.4: RGB-D-Caltech256 dataset. Classification accuracy variation for three classes of Table 5.1. In particular, from left to right: Accuracy variation against M, the number of training target domain samples; Accuracy variation against r, the dimensionality of T and S; Accuracy variation against the fraction of available auxiliary data; Convergence rate of the accuracy against the number of iterations of the learning procedure.

in [10], based on 5-fold cross-validation to select the best parameters and use them to retrain on the complete set. The main parameters to select are C, β, γ, and r, which is the number of columns of A. The C's and β's were searched in the range $\{10^{-3}, \cdots, 10^{3}\}$, the γ's in the range $\{0.1, 0.3, 0.5\}$. r was set by doing PCA on the mapped main view data (through $\phi(\cdot)$), and thresholding at 90% of the summation of the eigenvalues. In addition, for `DA-M2S` we set two parameters as indicated in [59], while for C and the others we look for those that maximize performance.

Performance: Average classification accuracy and standard deviation are reported. Testing is always done on the target domain data.

Object recognition: We evaluate the proposed approach for object recognition where we use the RGB-D Object dataset [11] as source domain, and the Caltech256 dataset [12] as target domain. We follow the same protocol outlined in [59], where we consider the 10 classes reported in Table 5.1, which are in common between the two datasets. Instances in the RGB-D Object are given as videos, and we uniformly sample frames every two seconds, obtaining 2056 training images. All the images of the 10 Caltech256 classes instead are used as unlabeled training target data.

Following [59], kernel descriptor (KDES) features [204], which perform well on the RGB-D Object dataset, are computed from the color and depth images to represent the main and the auxiliary views, respectively, and KDES features from the color images of the Caltech256 represent the target view. For each view we compute the Gradient KDES and the LBP KDES and we concatenate them. We set the vocabulary size to 1000, and use three level of pyramids.

For each of the 10 object classes, Table 5.1 and Table 5.2 shows the accuracies for the one-vs-all

binary classification with linear and Gaussian kernels, respectively. Here we randomly selected 50 positive and 50 negative training samples from the source domain, and the experiment was repeated 10 times. We observe that on average the multi-view based methods perform on par with the SVM, and the LUPI methods better exploit the information from the auxiliary view, but they all suffer from the lack of adaptation. The UDA methods perform better overall, highlighting the need to address the domain shift before taking advantage of the auxiliary view. In particular, we notice that LMIBDA, which does not use the auxiliary view, is an effective UDA approach. The last two columns address domain shift while leveraging the auxiliary view information, and show that the proposed LMIBDAPI provides state-of-the-art performance on this task.

Table 5.4 and Table 5.3 shows the classification accuracies for the multiclass classification case using linear and Gaussian kernels, respectively, where all the source samples are used for training. Even for this case, UDA methods improve upon the baseline SVM, and LMIBDA performs effectively, while LMIBDAPI confirms to have the best performance.

Figure 5.4 shows how the one-vs-all binary classification accuracy for three classes of Table 5.1 varies with respect to a number of parameters. The leftmost plot shows how the accuracy changes against the number M of training target domain samples. After a number of samples (about 200 in this case), the model saturates and additional samples will no more compensate for data shift. The second plot from the left shows that increasing r (i.e., the dimensionality of S and T), does not help beyond a certain limit (here between 60 and 70). Once it is reached, the model has enough capacity to extract all the necessary information for prediction. Beyond that limit the accuracy does not improve anymore and shows a noisy behavior. Choosing r below the limit reduces the capacity and thus prediction accuracy. The second plot from the right shows the accuracy variation against the fraction of available auxiliary data (or conversely, the fraction of missing auxiliary data). Note that handling missing auxiliary data is peculiar to our approach. The plot shows that at least 20% of missing auxiliary data is tolerated without performance drop. Finally, the rightmost plot shows

Table 5.3: RGB-D-Caltech256 dataset. Classification accuracies for the multi-class classification with Gaussian kernels. Main and auxiliary views are KDES features of the RGB and depth of the RGB-D Object dataset [11]. KDES features from the Caltech256 dataset [12] represent the target domain.

	UDA						UDA+LUPI	
SVM	SGF	LMK	SA	TCA	DIP	LMIBDA	DA-M2S	LMIBDAPI
18.23	19.41	19.69	19.83	25.07	25.47	27.23	29.47	**34.22**

	UDA				UDA+LUPI	
SVM	SGF	LMK	SA	LMIBDA	DA-M2S	LMIBDAPI
17.94	18.22	18.36	19.19	26.15	30.74	**33.66**

Table 5.4: RGB-D-Caltech-256 dataset. Classification accuracies for the multi-class classification with linear kernel. Main and auxiliary views are KDES features of the RGB and depth of the RGB-D Object dataset [11]. KDES features from the Caltech-256 dataset [12] represent the target domain.

Table 5.5: Office dataset. Classification accuracy for domain adaptation over the 31 categories of the Office dataset [13]. $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$ stand for Amazon, Webcam, and DSLR domain.

	SVM-s	SVM-t	LMK	HFA	GFK	SDASL	LMIBDA
$\mathcal{A} \to \mathcal{W}$	51.95	80.94	81.15	78.61	83.26	85.40	**86.10**
$\mathcal{A} \to \mathcal{D}$	54.92	82.90	82.31	83.71	82.72	**85.77**	85.31
$\mathcal{W} \to \mathcal{A}$	49.21	63.91	60.24	65.65	65.92	67.26	**67.41**
$\mathcal{W} \to \mathcal{D}$	83.26	81.91	82.26	86.10	84.28	86.18	**87.15**
$\mathcal{D} \to \mathcal{A}$	48.51	62.98	62.18	64.60	65.45	66.76	**66.82**
$\mathcal{D} \to \mathcal{W}$	80.35	82.65	83.45	81.69	82.69	**84.65**	83.36

the rate of convergence of the optimization procedure, which occurs monotonically. We found that no more than 10 iterations were normally enough to reach convergence, which is fairly good.

Table 5.5 shows the classification accuracy of the proposed approach for UDA without auxiliary data on the Office dataset [13], which contains 31 object classes for 3 domains: Amazon, Webcam, and DSLR, indicated as $\mathcal{A}$, $\mathcal{W}$, and $\mathcal{D}$, for a total of 4,652 images. The first domain consists of images downloaded from online merchants, the second consists of low resolution images acquired by webcams, the third consists of high resolution images collected with digital SLRs. The table notation $\mathcal{A} \to \mathcal{W}$ indicates that $\mathcal{A}$ was the source domain, and $\mathcal{W}$ the target. All the source data was used for training, whereas the target data was evenly split into two halves: one used for training and the other for testing. We used the 1000-way fc8 classification layer computed by DeCAF [45] as image features, and Gaussian kernels set up as detailed in [68]. We compared `LMIBDA` against `LMK`, the heterogeneous domain adaptation method (`HFA`) [205], the geodesic flow kernel method (`GFK`) [66], and against a recent semi-supervised domain adaptation method (`SDASL`) [68], which uses some labeled target data for training. The SVM trained on the source and on the target domain data, indicated as `SVM-s` and `SVM-t`, is also reported for reference. The main result is that even with this more popular domain adaptation dataset, the proposed approach, restricted to UDA only, has performance comparable to the state-of-the art.

	LUPI			UDA			UDA+LUPI	
SVM	SVM2k	KCCA	SVM+	LMK	SA	LMIBDA	DA-M2S	LMIBDAPI
64.89 ± 1.11	65.23 ± 11.37	63.23 ± 2.12	66.39 ± 1.23	63.45 ± 1.61	67.11 ± 1.46	67.29 ± 1.54	67.89 ± 1.32	**71.23 ± 1.23**

Table 5.6: EURECOM-LFW-a dataset. Classification accuracies for the male vs. female classification with linear kernel. Main and auxiliary views are Gradient-LBP features of the RGB and depth of the EURECOM dataset [14]. Gradient-LBP features from the LFW-a dataset [15] represent the target domain.

Gender recognition: We evaluate the proposed approach also for gender recognition where we use the RGB-D face dataset EURECOM [14] as source domain, and the RGB dataset Labeled Faces in the Wild-a (LFW-a) [15] as target domain. The EURECOM dataset consists of pairs of RGB and depth images from 196 females and 532 males captured with the Kinect sensor, and we removed the profile face images, which had only one manually annotated eye position. The LFW-a dataset contains images from 2,960 females and 10,184 males captured in uncontrolled conditions.

We resized the main, the auxiliary, and the target view face images to 120×105 pixels, and divide them into 8×7 non-overlapping subregions of 15×15 pixels. From each subregion of an image we extract the Gradient-LBP features, shown to be effective for gender recognition [14], and concatenate them into a single feature vector.

We perform a gender recognition experiment by combining the female source pairs with 196 randomly selected male source pairs to have a balanced gender representation. In addition, we randomly sample 3000 unlabeled target face images for training. The experiment is repeated 10 times, and the classification accuracies of all the methods are reported in Table 5.6 and Table 5.7 for linear and Gaussian kernels, respectively. The results show a pattern similar to the one found for object recognition in Table 5.1, Table 5.2, Table 5.4, and Table 5.3. One difference might be that in this experiment leveraging the auxiliary depth information seems to be as important as addressing the RGB domain shift. This is because the performance increase of the best LUPI methods is comparable to the performance increase of the best UDA methods. We also note that even here, `LMIBDA` confirms to be an effective UDA method by surpassing all the UDA and LUPI methods. Finally, although `DA-M2S` marginally improves by leveraging auxiliary information and addressing domain shift, the proposed `LMIBDAPI` provides a remarkable performance increase.

Table 5.7: EURECOM-LFW-a dataset. Classification accuracies for the male vs. female classification with Gaussian kernels. Main and auxiliary views are Gradient-LBP features of the RGB and depth of the EURECOM dataset [14]. Gradient-LBP features from the LFW-a dataset [15] represent the target domain.

	MV and LUPI			UDA						UDA+LUPI	
SVM	SVM2k	KCCA	SVM+	SGF	LMK	SA	TCA	DIP	LMIBDA	DA-M2S	LMIBDAPI
64.82 ± 1.35	67.15 ± 1.25	63.85 ± 1.34	67.31 ± 1.96	67.81 ± 1.45	64.88 ± 1.31	67.11 ± 1.45	65.24 ± 0.88	64.84 ± 4.80	68.11 ± 1.64	68.22 ± 1.41	**72.43 ± 1.34**

5.4 Conclusions

We developed an unsupervised domain adaptation approach for visual recognition when auxiliary information is available at training time. We extended the IB principle to IBDAPI, a new information theoretic principle that jointly handles the auxiliary view and the mismatch between the source and target distributions. We provided a modified version of IBDAPI based on risk minimization for learning explicitly any type of classifier, where training samples with missing auxiliary view can be handled seamlessly. We used this principle for deriving LMIBDAPI, a large-margin classifier with a fast optimization procedure in the primal space that converges in about 10 iterations. We performed experiments on object and gender recognition on a new target RGB domain by learning from a different RGB plus depth dataset. We observed that without using auxiliary data LMIBDA performs UDA with performance comparable with the state-of-the art. In addition, LMIBDAPI consistently outperformed the state-of-the-art, confirming its ability to carry the content of the auxiliary information over to a new domain.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, we explored several types of additional information that may be collected in an inexpensive manner during training of a visual classifier in order to improve its performance. Depending on where the additional information comes from, we showed that training a visual recognition model can be formulated as `domain adaptation (DA)`, `domain generalization (DG)`, `learning using privileged information (LUPI)`, and `domain adaptation with privileged information (DAPI)`.

Domain Adaptation and Generalization. We focused on a specific scenario for DA for when there are very few labeled target samples in training. We introduced two novel approaches. First, we developed a deep model in combination with the classification and contrastive semantic alignment (CCSA) loss to address few-shot domain adaptation. In this scenario, alignment and separation of semantic probability distributions is difficult because of the lack of data. We found that by reverting to point-wise surrogates of distribution distances and similarities provides an effective solution. Our extensive experiments show that our model converges very quickly as more labeled target samples per category are available. We have shown that the CCSA loss can be augmented to address the domain generalization problem without the need to change the basic model architecture.

Second, we followed the recent development in image generation and used adversarial learning for a novel few-shot domain adaptation. This is important because there is no other prior work addressing few-shot domain adaptation using adversarial learning. We found that by carefully

designing a training scheme whereby the typical binary adversarial discriminator is augmented to distinguish between four different classes, it is possible to effectively address the supervised adaptation problem. In addition, the approach has a high speed of adaptation, i.e. it requires an extremely low number of labeled target training samples, even one per category can be effective.

Learning Using Privileged Information. Privileged information has been shown to be effective in several computer vision applications. However, most of the works in this topic either modified a specific classifier in order to exploit privileged data or used a specific privileged data to improve the recognition task. In this dissertation, we provided a general framework that can use any type of classifier to exploit any privileged data. We focused on binary and multi-class SVM in our study and provided an optimization algorithm that is guaranteed to converge. Our method consistently outperformed the state-of-the-art LUPI methods.

Domain Adaptation with Privileged Information. There are very few studies on investigating the problem of covariate shift in presence of privileged data. Similar to our LUPI approach, we used the information bottleneck principal to develop a model for addressing DAPI. Our model is general in the sense that it allows the use of any classifier and any privileged data. We performed experiments on object and gender recognition on a new target RGB domain by learning from a different RGB plus depth dataset. Our method outperforms the state-of-the-art methods, confirming its ability to carry the content of the auxiliary information over to a new domain.

6.2 Future Work

6.2.1 Domain Adaptation

In Chapters 2 and 3, we proposed two models for supervised (few-shot) domain adaptation (SDA). However, we have not extended our models in presence of unlabeled target data (semi-supervised setting). This could be done by using an unsupervised domain adaptation (UDA) method in combination with our approaches.

As we discussed in Chapter 1, adversarial learning is very popular for UDA [89, 90, 88]. Those approaches use a binary discriminator to maximally confuse source and target distributions in the latent space. In Chapter 3, we discussed how we can use a multi-class discriminator to confuse the source and target samples in the latent space to address SDA. For future work, it would be

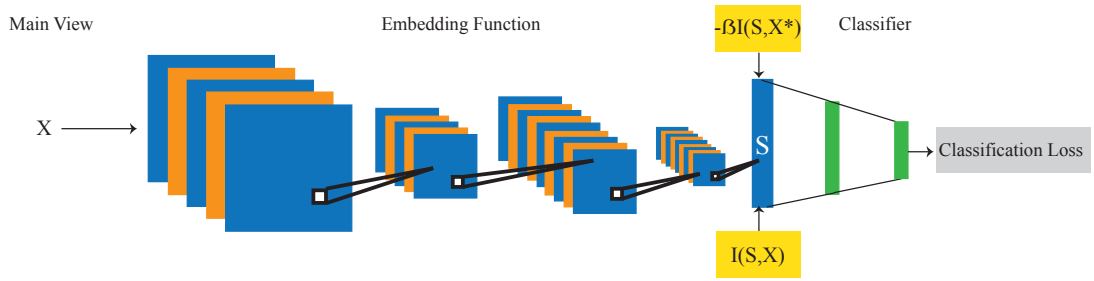

Figure 6.1: Deep Information Bottleneck Privileged Information. The regularizers are shown using yellow boxes.

interesting to investigate the use of two discriminators (one for addressing UDA and one for SDA) to address the semi-supervised setting.

6.2.2 Domain Generalization

As we discussed, adversarial learning can be used to maximally confuse source and target distributions in the latent space. By following the same idea, adversarial learning may be very effective to confuse several source domains in the latent space to address the domain generalization problem. This idea could also be very effective for unsupervised domain generalization (where there is no label information for some source domains).

6.2.3 Deep Information Bottleneck for Visual Recognition

In Chapters 4 and 5, we introduced the general frameworks for information bottleneck (IB) learning for visual recognition. Since computing mutual information is difficult and is limited to some special cases, IB is hard to use in deep networks. Recently some techniques have been developed to address this issue [206, 207]. [206] proposed a method to perform IB in more general domains by defining an upper bound on the IB objective, derived using a non-parametric estimator of mutual information and a variational approximation. [207] proposed a method to parameterize the information bottleneck model using a neural network and leverage the reparameterization trick for efficient training.

For future work, one may be interested in using deep classifiers in (4.7) and (5.6) instead of SVM. To learn the parameters of the deep classifiers, we can build on techniques from [206, 207]. Specifically, focusing on (4.7), in Chapter 4, we first assumed that there is a linear mapping between

the main view X and the latent variable S and then used the kernel trick for non-linear mapping. For future work, one may use deep embedding functions (containing convolutional and fully connected layers) in order to embed X onto S. See Figure 6.1. In terms of implementation, the first and second terms in (4.7) can be seen as the regularizers on the last layer of the embedding function as shown if Figure 6.1. A discussion is valid also for the Privilege Information and Domain Adaptation framework (5.6).

References

[1] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 1735–1742.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[3] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *NIPS*, 2006.

[4] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision–ECCV 2016 Workshops*. Springer, 2016, pp. 443–450.

[5] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[8] N. Tishby, F. Pereira, and W. Bialek, "The information bottleneck method," in *Allerton Conference on Communication, Control, and Computing*, 1999, pp. 368–377.

[9] Z. Wang and Q. Ji, "Classifier learning with hidden information," in *CVPR*, June 2015, pp. 4969–4977.

[10] V. Sharmanska, N. Quadrianto, and C. Lampert, "Learning to rank using privileged information," in *IEEE ICCV*, 2013, pp. 825–832.

[11] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *IEEE ICRA*, 2011.

[12] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep., 2007.

[13] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *ECCV*, 2010, pp. 213–226.

[14] T. Huynh, R. Min, and J. Dugelay, "An efficient LBP-based descriptor for facial depth images applied to gender recognition using RGB-D face data," in *ACCV Workshops*, 2012, pp. 133–145.

[15] L. Wolf, T. Hassner, and Y. Taigman, "Effective unconstrained face recognition by combining multiple descriptors and learned background statistics," *IEEE TPAMI*, vol. 33, no. 10, pp. 1978–1990, 2011.

[16] W. T. Freeman, K.-i. Tanaka, J. Ohta, and K. Kyuma, "Computer vision for computer games," in *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*. IEEE, 1996, pp. 100–105.

[17] D.-W. Sun, *Computer vision technology in the food and beverage industries*. Elsevier, 2012.

[18] M. Moosaei, M. J. Gonzales, and L. D. Riek, "Naturalistic pain synthesis for virtual patients," in *Intelligent Virtual Agents*. Springer, 2014, pp. 295–309.

[19] M. Moosaei, C. J. Hayes, and L. D. Riek, "Performing facial expression synthesis on robot faces: A real-time software system," 2015.

[20] M. Moosaei, S. K. Das, D. O. Popa, and L. D. Riek, "Using facially expressive robots to calibrate clinical pain perception," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. ACM, 2017, pp. 32–41.

[21] S. Shukla, A. Lakhmani, and A. K. Agarwal, "Approaches of artificial intelligence in biomedical image processing: A leading tool between computer vision & biological vision," in *Advances in Computing, Communication, & Automation (ICACCA)(Spring), International Conference on*. IEEE, 2016, pp. 1–6.

[22] S. Motiian, P. Pergami, K. Guffey, C. A. Mancinelli, and G. Doretto, "Automated extraction and validation of childrens gait parameters with the kinect," *Biomedical engineering online*, vol. 14, no. 1, p. 112, 2015.

[23] F. Siyahjani and G. Doretto, *Learning a Context Aware Dictionary for Sparse Representation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 228–241.

[24] D. Alhelal, K. A. Aboalayon, M. Daneshzand, and M. Faezipour, "Fpga-based denoising and beat detection of the ecg signal," in *Systems, Applications and Technology Conference (LISAT), 2015 IEEE Long Island*. IEEE, 2015, pp. 1–5.

[25] S. Motiian, K. Feng, H. Bharthavarapu, S. Sharlemin, and G. Doretto, "Pairwise kernels for human interaction recognition," in *International Symposium on Visual Computing*. Springer, 2013, pp. 210–221.

[26] F. Siyahjani, S. Motiian, H. Bharthavarapu, S. Sharlemin, and G. Doretto, "Online geometric human interaction segmentation and recognition," in *Multimedia and Expo (ICME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.

[27] F. Siyahjani, R. Almohsen, S. Sabri, and G. Doretto, "A supervised low-rank method for learning invariant subspaces," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4220–4228.

[28] M. Daneshzand, M. Faezipour, and B. D. Barkana, "Computational stimulation of the basal ganglia neurons with cost effective delayed gaussian waveforms," *Frontiers in computational neuroscience*, vol. 11, p. 73, 2017.

[29] S. Motiian, F. Siyahjani, R. Almohsen, and G. Doretto, "Online human interaction detection and recognition with multiple cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 3, pp. 649–663, 2017.

[30] F. Siyahjani, "Subspace representations and learning for visual recognition," Ph.D. dissertation, West Virginia University, 2017.

[31] M. D. Zand, A. H. Ansari, C. Lucas, and R. A. Z. Zoroofi, "Risk assessment of coronary arteries heart disease based on neuro-fuzzy classifiers," in *Biomedical Engineering (ICBME), 2010 17th Iranian Conference of*. IEEE, 2010, pp. 1–4.

[32] A. Torfi, N. M. Nasrabadi, and J. Dawson, "Text-independent speaker verification using 3d convolutional neural networks," *arXiv preprint arXiv:1705.09422*, 2017.

[33] M. Iranmanesh, A. Dabouei, H. Kazemi, and N. M. Nasrabadi, "Deep cross polarimetric thermal-to-visible face recognition," in *Biometrics (ICB), 2018 International Conference on*. IEEE, 2018.

[34] A. Torfi, S. M. Iranmanesh, N. Nasrabadi, and J. Dawson, "3d convolutional neural networks for cross audio-visual matching recognition," *IEEE Access*, vol. 5, pp. 22 081–22 091, 2017.

[35] A. Dabouei, H. Kazemi, M. Iranmanesh, and N. M. Nasrabadi, "Fingerprint distortion rectification using deep convolutional neural networks," in *Biometrics (ICB), 2018 International Conference on*. IEEE, 2018.

[36] J. J. Donai, S. Motiian, and G. Doretto, "Automated classification of vowel category and speaker type in the high-frequency spectrum," *Audiology research*, vol. 6, no. 1, 2016.

[37] H. Kazemi, M. Iranmanesh, A. Dabouei, and N. M. Nasrabadi, "Facial attributes guided deep sketch-to-photo synthesis," in *Applications of Computer Vision (WACV), 2018 IEEE Workshop on*. IEEE, 2018.

[38] A. Broumand, M. S. Esfahani, B.-J. Yoon, and E. R. Dougherty, "Discrete optimal bayesian classification with error-conditioned sequential sampling," *Pattern Recognition*, vol. 48, no. 11, pp. 3766–3782, 2015.

[39] H. Kazemi, S. Soleymani, A. Dabouei, M. Iranmanesh, and N. M. Nasrabadi, "Attribute-centered loss for soft-biometrics guided face sketch-photo recognition," *arXiv preprint arXiv:1804.03082*, 2018.

[40] A. Broumand and T. Hu, "A length bias corrected likelihood ratio test for the detection of differentially expressed pathways in rna-seq data," in *Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on*. IEEE, 2015, pp. 1145–1149.

[41] S. Soleymani, A. Dabouei, H. Kazemi, J. Dawson, and N. M. Nasrabadi, "Multi-level feature abstraction from convolutional neural networks for multimodal biometric identification," in *Pattern Recognition (ICPR), 2018 24th International Conference on*. IEEE, 2018, pp. 1–8.

[42] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[43] J. Weston and C. Watkins, "Multi-class support vector machines," Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May, Tech. Rep., 1998.

[44] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[45] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: a deep convolutional activation feature for generic visual recognition," in *arXiv:1310.1531*, 2013.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[47] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, 2015.

[49] F. Lauer and G. Bloch, "Incorporating prior knowledge in support vector machines for classification: A review," *Neurocomputing*, vol. 71, no. 7–9, pp. 1578–1594, 2008.

[50] M. Wulfmeier, D. Rao, and I. Posner, "Incorporating human domain knowledge into large scale cost function learning," *arXiv preprint arXiv:1612.04318*, 2016.

[51] Ç. Gülçehre and Y. Bengio, "Knowledge matters: Importance of prior information for optimization," *Journal of Machine Learning Research*, vol. 17, no. 8, pp. 1–32, 2016.

[52] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[53] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006, pp. 120–128.

[54] G. Blanchard, G. Lee, and C. Scott, "Generalizing from several related classification tasks to a new unlabeled sample," in *Advances in neural information processing systems*, 2011, pp. 2178–2186.

[55] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 1521–1528.

[56] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Networks*, vol. 22, no. 5–6, pp. 544–557, 2009.

[57] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[58] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, 2017.

[59] L. Chen, W. Li, and D. Xu, "Recognizing RGB images by learning from RGB-D data," in *CVPR*, June 2014, pp. 1418–1425.

[60] B. Gong, "Kernel methods for unsupervised domain adaptation," Ph.D. dissertation, University of Southern California, 2015.

[61] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[62] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *ICCV*, 2015.

[63] P. Koniusz, Y. Tas, and F. Porikli, "Domain adaptation by mixture of alignments of second-or higher-order scatter tensors," *arXiv preprint arXiv:1611.08195*, 2016.

[64] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *European Conference on Computer Vision*. Springer, 2016, pp. 597–613.

[65] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks." in *ICML*, 2015, pp. 97–105.

[66] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2066–2073.

[67] Y. Guo and M. Xiao, "Cross language text classification via subspace co-regularized multi-view learning," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[68] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei, "Semi-supervised domain adaptation with subspace learning for visual recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[69] J. Deng, Z. Zhang, F. Eyben, and B. Schuller, "Autoencoder-based unsupervised domain adaptation for speech emotion recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1068–1072, 2014.

[70] S. Sun, B. Zhang, L. Xie, and Y. Zhang, "An unsupervised deep domain adaptation approach for robust speech recognition," *Neurocomputing*, 2017.

[71] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta, "The curious robot: Learning visual representations via physical interactions," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–18.

[72] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1–2, pp. 151–175, 2009.

[73] J. Farquhar, D. R. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmak, "Two view learning: SVM-2K, theory and practice," in *NIPS*, 2006.

[74] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *ICCV*, Sept 2009, pp. 606–613.

[75] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *ICCV*, 2009.

[76] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *ICML*, 2011.

[77] C. Xu, D. Tao, and C. Xu, "Large-margin multi-view information bottleneck," *IEEE TPAMI*, vol. 36, no. 8, pp. 1559–1572, 2014.

[78] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," in *Advances in Neural Information Processing Systems*, 2017, pp. 6673–6683.

[79] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Information bottleneck learning using privileged information for visual recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1496–1505.

[80] S. Motiian and G. Doretto, "Information bottleneck domain adaptation with privileged information for visual recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–647.

[81] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1785–1792.

[82] M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, and P. S. Yu, "Transfer sparse coding for robust image representation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 407–414.

[83] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *IEEE ICCV*, 2013, pp. 769–776.

[84] A. Bergamo and L. Torresani, "Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach," in *Advances in Neural Information Processing Systems*, 2010, pp. 181–189.

[85] Y. Aytar and A. Zisserman, "Tabula rasa: Model transfer for object category detection," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2252–2259.

[86] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[87] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[88] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 469–477.

[89] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[90] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," *arXiv preprint arXiv:1704.01705*, 2017.

[91] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 1875–1882.

[92] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[93] M. Lapin, M. Hein, and B. Schiele, "Learning using privileged information: SVM+ and weighted SVM," *Neural Networks*, vol. 53, pp. 95–108, 2014.

[94] W. Li, L. Niu, and D. Xu, "Exploiting privileged information from web data for image categorization," in *ECCV*, 2014, pp. 437–452.

[95] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba *et al.*, "Dataset issues in object recognition," in *Toward category-level object recognition*. Springer, 2006, pp. 29–48.

[96] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars, "A deeper look at dataset bias," in *German Conference on Pattern Recognition*. Springer, 2015, pp. 504–516.

[97] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *IEEE ICCV*, 2011, pp. 999–1006.

[98] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *IEEE ICCV*, 2013, pp. 2960–2967.

[99] T. Tommasi, M. Lanzi, P. Russo, and B. Caputo, "Learning the roots of visual domain shift," in *Computer Vision–ECCV 2016 Workshops*. Springer, 2016, pp. 475–482.

[100] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *NIPS*, 2006.

[101] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," 2009.

[102] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye, "A two-stage weighting framework for multi-source domain adaptation," in *Advances in neural information processing systems*, 2011, pp. 505–513.

[103] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," *arXiv preprint arXiv:1611.02200*, 2016.

[104] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[105] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[106] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[107] W. Deng, L. Zheng, G. Kang, Y. Yang, Q. Ye, and J. Jiao, "Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person re-identification," *arXiv preprint arXiv:1711.07027*, 2017.

[108] A. Anoosheh, E. Agustsson, R. Timofte, and L. Van Gool, "Combogan: Unrestrained scalability for image domain translation," *arXiv preprint arXiv:1712.06909*, 2017.

[109] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation." in *ICML (1)*, 2013, pp. 10–18.

[110] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[111] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[112] J. Blitzer, S. Kakade, and D. Foster, "Domain adaptation with coupled subspaces," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 173–181.

[113] R. Li and T. Zickler, "Discriminative virtual views for cross-view action recognition," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2855–2862.

[114] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE TNN*, vol. 22, no. 2, pp. 199–210, 2011.

[115] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, 1994, pp. 737–744.

[116] B. Kumar, G. Carneiro, I. Reid *et al.*, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5385–5394.

[117] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang, "A siamese long short-term memory architecture for human re-identification," in *European Conference on Computer Vision*. Springer, 2016, pp. 135–153.

[118] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *arXiv preprint arXiv:1603.06432*, 2016.

[119] J. Yang, R. Yan, and A. G. Hauptmann, "Adapting svm classifiers to data with shifted distributions," in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 69–76.

[120] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank, "Domain transfer svm for video concept detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1375–1381.

[121] C. J. Becker, C. M. Christoudias, and P. Fua, "Non-linear domain adaptation with boosting," in *Advances in Neural Information Processing Systems*, 2013, pp. 485–493.

[122] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, 2006.

[123] Y. Yang and T. Hospedales, "Zero-shot domain adaptation via kernel regression on the grassmannian," *arXiv preprint arXiv:1507.07830*, 2015.

[124] K.-C. Peng, Z. Wu, and J. Ernst, "Zero-shot deep domain adaptation," *arXiv preprint arXiv:1707.01922*, 2017.

[125] A. A. Deshmukh, S. Sharma, J. W. Cutler, and C. Scott, "Multiclass domain generalization."

[126] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter component analysis: A unified framework for domain adaptation and domain generalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[127] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2551–2559.

[128] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, "Undoing the damage of dataset bias," in *European Conference on Computer Vision*. Springer, 2012, pp. 158–171.

[129] C. Fang, Y. Xu, and D. N. Rockmore, "Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias," in *International Conference on Computer Vision*, 2013.

[130] Z. Xu, W. Li, L. Niu, and D. Xu, "Exploiting low-rank structure from latent domains for domain generalization," in *ECCV*, 2014, pp. 628–643.

[131] L. Niu, W. Li, and D. Xu, "Multi-view domain generalization for visual recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4193–4201.

[132] L. Niu, W. Li, D. Xu, and J. Cai, "An exemplar-based multi-view domain generalization framework for visual recognition," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[133] J. Hoffman, S. Gupta, and T. Darrell, "Learning with side information through modality hallucination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 826–834.

[134] Z. Wang, X. Wang, and Q. Ji, "Learning with hidden information," in *ICPR*, Aug 2014, pp. 238–243.

[135] Z. Wang, T. Gao, and Q. Ji, "Learning with hidden information using a max-margin latent variable model," in *ICPR*, Aug 2014, pp. 1389–1394.

[136] D. Pechyony and V. Vapnik, "On the theory of learning with privileged information," in *NIPS*, 2010.

[137] J. Donahue and K. Grauman, "Annotator rationales for visual recognition," in *ICCV*, Nov 2011, pp. 1395–1402.

[138] J. Tang, Y. Tian, P. Zhang, and X. Liu, "Multiview privileged support vector machines," *IEEE transactions on neural networks and learning systems*, 2017.

[139] N. Sarafianos, C. Nikou, and I. A. Kakadiaris, "Predicting privileged information for height estimation," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 3115–3120.

[140] S. Wang, D. Tao, and J. Yang, "Relative attribute svm+ learning for age estimation," *IEEE transactions on cybernetics*, vol. 46, no. 3, pp. 827–839, 2016.

[141] S. You, C. Xu, Y. Wang, C. Xu, and D. Tao, "Privileged multi-label learning," *arXiv preprint arXiv:1701.07194*, 2017.

[142] B. Mahasseni and S. Todorovic, "Regularizing long short term memory with 3d human-skeleton sequences for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3054–3062.

[143] Z. Shi and T.-K. Kim, "Learning and refining of privileged information-based rnns for action recognition from depth sequences," *arXiv preprint arXiv:1703.09625*, 2017.

[144] Z. Luo, L. Jiang, J.-T. Hsieh, J. C. Niebles, and L. Fei-Fei, "Graph distillation for action detection with privileged information," *arXiv preprint arXiv:1712.00108*, 2017.

[145] V. Ferrari and A. Zisserman, "Learning visual attributes," in *NIPS*, 2007.

[146] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *CVPR*, June 2009, pp. 1778–1785.

[147] L. Torresani, M. Szummer, and A. Fitzgibbon, "Efficient object category recognition using classemes," in *ECCV*, 2010, pp. 776–789.

[148] Q. Li, J. Wu, and Z. Tu, "Harvesting mid-level visual concepts from large-scale internet images," in *CVPR*, June 2013, pp. 851–858.

[149] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE TPAMI*, vol. 32, no. 9, pp. 1627–1645, Sept 2010.

[150] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE TPAMI*, vol. 29, no. 10, pp. 1848–1852, Oct 2007.

[151] N. Slonim and N. Tishby, "Agglomerative information bottleneck," in *NIPS*, 1999.

[152] G. Chechik and N. Tishby, "Extracting relevant structures with side information," in *NIPS*, 2002.

[153] L. Wolf and N. Levy, "The SVM-Minus similarity score for video face recognition," in *CVPR*, June 2013, pp. 3523–3530.

[154] J. Chen, X. Liu, and S. Lyu, "Boosting with side information," in *ACCV*, 2012, pp. 563–577.

[155] J. Feyereisl, S. Kwak, J. Son, and B. Han, "Object localization based on structural SVM using privileged information," in *NIPS*, 2014.

[156] S. Fouad, P. Tino, S. Raychaudhury, and P. Schneider, "Incorporating privileged information through metric learning," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 7, pp. 1086–1098, July 2013.

[157] X. Xu, W. Li, and D. Xu, "Distance metric learning using privileged information for face verification and person re-identification," *IEEE Trans. on Neural Networks and Learning Systems*, 2015.

[158] M. Xu, R. Jin, and Z.-H. Zhou, "Speedup matrix completion with side information: Application to multi-label learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2301–2309.

[159] X. Yang, M. Wang, and D. Tao, "Person re-identification with metric learning using privileged information," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 791–805, 2018.

[160] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "Active privileged learning of human activities from weakly labeled samples," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3036–3040.

[161] Q. Zhang, G. Hua, W. Liu, Z. Liu, and Z. Zhang, "Can visual recognition benefit from auxiliary information in training?" in *ACCV*, 2014, pp. 65–80.

[162] Q. Zhang and G. Hua, "Multi-view visual recognition of imperfect testing data," in *ACM MM*, 2015, pp. 561–570.

[163] W. Li, L. Chen, D. Xu, and L. Van Gool, "Visual recognition in rgb images and videos by learning from rgb-d data," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[164] N. Sarafianos, M. Vrigkas, and I. A. Kakadiaris, "Adaptive svm+: Learning with privileged information for domain adaptation," *arXiv preprint arXiv:1708.09083*, 2017.

[165] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[166] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[167] H. Wang, W. Wang, C. Zhang, and F. Xu, "Cross-domain metric learning based on information theory." in *AAAI*, 2014, pp. 2099–2105.

[168] Z. Ding and Y. Fu, "Robust transfer metric learning for image classification," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 660–670, 2017.

[169] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[170] B. Fernando, T. Tommasi, and T. Tuytelaarsc, "Joint cross-domain classification and subspace learning for unsupervised adaptation," *Pattern Recogition Letters*, 2015.

[171] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[172] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[173] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[174] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky, "Exploiting hierarchical context on a large database of object categories," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 129–136.

[175] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 833–840.

[176] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936.

[177] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[178] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[179] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[180] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. JMLR. org, 2016, pp. 1060–1069.

[181] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," in *Advances in Neural Information Processing Systems*, 2016, pp. 217–225.

[182] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint arXiv:1612.03242*, 2016.

[183] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[184] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[185] N. Slonim, N. Friedman, and N. Tishby, "Multivariate information bottleneck," *Neural Computation*, vol. 18, no. 8, pp. 1739–1789, 2006.

[186] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley and Sons, Inc., 1991.

[187] C. J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[188] Z. Yang, H. Zhang, Z. Yuan, and E. Oja, "Kullback-Leibler divergence for nonnegative matrix factorization," in *ICANN*, 2011, pp. 250–257.

[189] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[190] T. Zhou, D. Tao, and X. Wu, "NESVM: A fast gradient method for support vector machines," in *ICDM*, Dec 2010, pp. 679–688.

[191] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Mathematical Programming*, vol. 141, no. 1–2, pp. 349–382, 2013.

[192] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[193] ——, "Training linear svms in linear time," in *KDD*, 2006.

[194] L. Liang and V. Cherkassky, "Connection between svm+ and multi-task learning," in *IJCNN*, 2008, pp. 2048 – 2054.

[195] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Computation*, vol. 16, p. 26392664, 2004.

[196] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *IEEE ICCV*, 2011.

[197] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[198] ChaLearn, "ChaLearn gesture dataset (CGD2011)," California, 2011.

[199] C. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE TPAMI*, vol. 36, no. 3, pp. 453–465, March 2014.

[200] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *JMLR*, vol. 2, pp. 265–292, 2001.

[201] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[202] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, and C.-J. Wang, X.-R.and Lin, "LIBLINEAR: A library for large linear classification," *JMLR*, no. 9, pp. 1871–1874, 2008.

[203] B. Gong, K. Grauman, and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation," in *ICML*, 2013.

[204] L. Bo, X. Ren, and D. Fox, "Depth kernel descriptors for object recognition," in *IROS*, 2011.

[205] L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for heterogeneous domain adaptation," in *Proceedings of the International Conference on Machine Learning*. Edinburgh, Scotland: Omnipress, June 2012, pp. 711–718.

[206] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear information bottleneck," *arXiv preprint arXiv:1705.02436*, 2017.

[207] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.